

Simulation of Smart Homes controlled with Intelligent Plugs

by

Victor Gil Garcia

30/05/2014

A dissertation submitted in partial fulfilment of
the requirements for the degree of
MASTER OF ENGINEERING in Computer Science

from the School of Electronics, Electrical Engineering and Computer Science,
Queen's University of Belfast

ACKNOWLEDGEMENTS

Professor Weiru Liu, Queen's University Belfast

ABSTRACT

This is a research project in the field of Smart Homes with the aim to develop a Smart Home Environment using Multi-Agent system negotiation and Rule-Based protocols. The simulation will be developed using the program Netbeans (Java) and Prolog. It will permit to control the status of the devices via Intelligent Plugs and display alerts. An Intelligent Plug is an appliance that goes between the plug and the socket, and the user can switch on and off using Wi-Fi connection.

The objective is to reduce the power consumption of household appliances using intelligent-adaptive algorithms, to develop a user-friendly interface and also to gain a better command over them. The developed application will permit the interaction of the user with it, enhancing the user's comfort at its finest. Once the expert system has been implemented, a Multi-Agent environment will be developed involving five different houses. It will have two different negotiation protocols: one to control the heat of the five houses and the other to control the total energy consumption of the neighbourhood.

Table of contents

Acknowledgements	1
Abstract	2
1. Problem specification	8
1.1 Introduction	8
1.1.1 General introduction	8
1.1.2 Smart houses	9
1.1.2.1 Smart meters	9
1.1.2.2 Expert system	10
1.2 Problem description	10
1.3 State of current search and development	11
1.3.1 Research in smart meters	11
1.3.2 Research in expert and multi-agent systems	12
1.4 Rules and expert system	13
1.4.1 Expert systems	13
1.4.2 Rules	14
1.5 Multi-agent negotiation	15
1.5.1 Negotiation parts	15
1.5.1.1 Agents	15
1.5.1.2 Negotiation objects	16
1.5.1.3 Negotiation protocols	16
1.6 Simulation enviroment	16
1.7 User interface	17
1.8 System especification	17
1.9 Expected outcome	17
2. Specification	18
2.1 Platform	18

2.1.1 Netbeans	18
2.2 Programming languages	18
2.2.1 Prolog	18
2.2.2 Java	19
2.3 External software	19
2.3.1 Swi-Prolog	19
2.3.2 JDK	19
3. Design	20
3.1 System functions	20
3.2 User requirements	20
3.3 Interface design	21
3.3.1 Devices	22
4. Methods: research methodologies	23
4.1 Expert system	23
4.1.1 Rule matching	23
4.1.2 Fuzzy matching	23
4.1.3 Weighting	24
4.1.4 Accuracy	24
4.1.5 Score formula	25
4.2 Multi-agent negotiation	26
4.2.1 Heat control system	26
4.2.1.1 Agents	26
4.2.1.2 Negotiation objects	26
4.2.1.3 Negotiation protocol	27
4.2.2 One-bid system	28
4.2.2.1 Agents	28
4.2.2.2 Negotiation objects	28
4.2.2.3 Negotiation protocol	29

5. Implementation	32
5.1 Expert system	32
5.1.1 Rules	32
5.1.2 Expert system code.....	38
5.2 Heat control.....	39
5.2.1 Phases of the negotiation protocol	39
5.2.2 Negotiation 2	40
5.3 Control net negotiation.....	42
5.3.1 Phases of the control net	42
5.3.2 Negotiation 1	43
6. Results and analysis	45
6.1 Analysis of the rule based system.....	45
6.1.1 Situation 1	45
6.1.2 Situation 2	46
6.1.3 Situation 3	46
6.1.4 Analysis in further work of the rule-based system.....	47
6.2 Analysis of the heat control negotiation	48
6.2.1 Bids analysys.....	48
6.2.2 Satisfaction analysis	49
6.3 Analysis of control net protocol	50
6.3.1 Independence between agents and bids	50
6.3.2 Importance of the preferences of the agents	52
7. Conclusions.....	53
8. Further work.....	54
9. References	55
9.1 Books and research papers	55
9.2 Internet sources	56
10. Appendices.....	58

10.1 Java implementation of the expert system	58
10.1.1 Initialize variables	58
10.1.2 Button functions	59
10.1.3 Step function	60
10.1.4 Calculate energy functions	62
10.1.5 State of rules function.....	63
10.2 Prolog implementation of the rule-set	64
10.2.1 General rule	64
10.2.2 Rule-set	64
10.2.3 Decision function	65
10.2.4 Other functions	66
10.3 Java implementation of the heat control system	67
10.3.1 Start negotiation function	67
10.3.2 Satisfaction functions and desired heat.....	68
10.3.3 Negotiation phase functions	68
10.3.4 Bid functions	71
10.4 Java implementation of the one-bid system.....	74
10.4.1 Main function	74
10.4.2 Negotiation function.....	75
10.4.3 Device functions.....	77
10.4.4 Bid functions	78
10.4.5 Solution functions.....	81
10.5 Screen captures of the simulation	82
10.5.1 Screen capture of the main simulator.....	82
10.5.2 Screen capture of the heat control window	84
10.5.3 Screen capture of the simulation during the one-bid negotiation	84
10.6 User's guide	85
10.6.1 General information	85

10.6.2 Rule-set guide.....	86
10.6.3 Heat control guide	86
10.6.4 One-bid system guide	87
10.7 Data for the graphs displayed	88
10.7.1 Data for the heat control system analysis	88
10.7.2 Data for the one-bid system analysis	91
10.8 Minutes of project supervision meeting.....	95

1. PROBLEM SPECIFICATION

1.1 INTRODUCTION

1.1.1 GENERAL INTRODUCTION

From the origins of mankind, energy has been a constant necessity. It started as a need to solve heating and mechanical problems and it evolved from there, through the industrial revolution and the first steps of electricity, to our current energy crisis.

Energy sources can be classified in two types: nonrenewal and renewal. Nonrenewal sources are obtained from the earth and can be exhausted. On one side, fossil fuels, a nonrenewal source that has little life left as it is rapidly depleted, are consumed at a rate 100.000 times that at which they are formed [19]. On the other side, renewal resources are way beyond giving its full potential and nowadays it is impossible to rely only on them. And also nuclear, which implies high efficiency but also a soaring damage to the environment and risk to human life.

In our days, the consumption habits of modern lifestyles are vastly increasing the consumption of energy. This non-stopping energy overuse is leading us to the biggest energy crisis in the human history.

In the UK the household electricity consumption is approximately 4,200 kWh per year [20].

Usage	Lightning & appliances	Space Heating	Water Heating	Cooking
Percentage of total use	75%	14%	6%	5%

Table: Initial breakdown of domestic demand [21]

On average UK households spend between £45 and £80 per year each powering appliances left on standby mode or not in use [22]. This is energy used by certain appliances when they are not in use and not switched off at the plug.

1.1.2 SMART HOUSES

This urge of a better house energy control has led to the development of Smart Houses or Home automation. This technology is based on using computers and information resources to control, among other features, home appliances. The systems can range from simple remote control of devices through complex computer based networks that take decisions by themselves.

1.1.2.1 SMART METERS

The first step in converting a house in a Smart environment would be to take control of the electrical switches and the devices that are plugged in. Electronic meters were the first assumption towards the development of Smart environments.

Electronic meters for electricity (Smart meters) are increasing its use in private homes all over the world. In the market there are two main types of smart plugs: Smart Plug Strip and Smart Plug Switch.

Smart Plug Strips use a Master plug to control the other devices. When the user turn off manually the device connected to the Master plug the Smart plug strip will shut down the slave devices automatically. For example: when the user turn off the PC, it turns off the speakers, monitor and printer. Smart Plug Switches, as seen in Figure 1.1, are a one-plug device that control one device per Smart plug and have Wi-Fi connection. The user can get an absolute power of the device powered to the plug. Scheduled functions can be implemented, such as turning on or off, and it can be managed in real time using the Wi-Fi connection.



Fig. 1.1. Smart Plug Switch. Edimax SP-1101W [23]

1.1.2.2 EXPERT SYSTEM

Once the electrical consumption is measured, to achieve the status of a Smart House it needs to be controlled with an algorithm more complex than simple Smart Plugs. In this part, the research led to the implementation of expert systems, allowing the system to have some rules that follow its train of thought. The last level is the addition of communication protocols between different agents (different houses) in order to achieve a common goal. This is reached with the use of Multi-Agent methods of communication and negotiation.

1.1 PROBLEM DESCRIPTION

The original problem described in the Specification Document was simpler than this actual problem description. The aim of the project is the same, to merge the ideas of Smart Plugs and Smart Strips to intelligently control the electrical consumption of five houses in the neighbourhood.

The main objective is to measure the energy consumption of each house, the total energy consumption of the five houses and to control the status of the devices that are plugged in the network. Each house will be developed as an expert system with an adaptive algorithm that controls the devices. The user will have the possibility to interact with them and the simulation will respond and adapt to this new status.

The user will be able to schedule turning on and off devices (i.e. lights off during the morning when sun light can be used). To establish schedules it will be necessary to control the time of the day during the simulation. And also, connections between the devices will be added to simulate a Smart Strip behaviour, which means that if one of the devices in the master slot of the plug is turned off, the others need to be shut down.

The second main objective is the development of a Multi-Agent negotiation protocol between the five houses. There will be two different negotiation protocols: a One-Bid system to control the total electric consumption of the neighbourhood and a second negotiation system to control the heating of the houses. The goal of the first one is to keep the total energy consumption under a certain level allowing the users to interact between themselves if one of them needs to turn on a device and the total energy consumption has reached its limit. If the negotiation succeeds the demanding agent will turn on his device and the offering agent will turn off his.

The second negotiation system will be used to control the temperature of each house in the neighbourhood and to try to maintain an acceptable level of satisfaction amongst them. The agent with the lowest satisfaction level will start the negotiation. Then, he will try to reach a common agreement with the other agents so they lower their house temperature so he can raise his temperature without surpassing the heating limit for the five houses.

1.2 STATE OF CURRENT SEARCH AND DEVELOPMENT

Several projects and research have been done in this field. Some of them are based on the same ideas as this project, but one of the differences is that none of them include all the aspects this project will include. Here they will be listed, explained and compared to the actual research.

1.3.1 RESEARCH IN SMART METERS

An example of an Intelligent Plug could be the *nPlug* [9]: A smart plug that sits between the wall socket and deferrable loads such as water heaters, washing machines, and electric vehicles. Combining real-time sensing and analytics to balance the on and off periods.

Furthermore, smart meters are being used in research to optimize different systems in a house, an example could be the paper on *Optimization of a Heating System Using Smarts Meters & Dynamic Pricing* [10]. This work describes the process of optimizing a heating system while considering dynamic pricing. A cost function is created combining the cost of electricity used for heating and the cost off the loss of comfort. The problem that a household will find with this project is that it is difficult to create an accurate model that represents their home. In our project will focus only on the lighting and devices connected so that it is easier to have a complete and accurate command of it.

Complex algorithms are being implemented to control the home energy as in *An Algorithm for Intelligent Home Energy Management and Demand Response Analysis* [13], which manages household loads according to their pre-established priority and guarantees the total power consumption below certain levels. This paper presents an intelligent Home Energy Management (HEM) algorithm for managing high power consumption household appliances with simulation for demand response analysis. Simulation results show that the proposed algorithm can effectively control and manage the appliance operation to keep the total household consumption below a specified demand limit. The difference between this paper and the project's research is that, in the

project the comfort level will never be sacrificed and that the devices will be managed with more algorithms that just keeping it under the demand level.

1.3.2 RESEARCH IN EXPERT AND MULTI-AGENT SYSTEMS

Rule-Based systems have been used in multiple research to control all kinds of behaviours. In *System Integration of GIS and a Rule-Based Expert System for Urban Mapping* [5], the expert system uses a question-and-answer method for spatial data mapping. Rule-Based systems are the most common artificial intelligence technique to information systems, in this case, a geographic information system.

There have also been implemented in smart home environments already, as in *A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation* [15]. In that paper the researchers implemented a multi-agent mechanism in a Smart House and they simulated it using the Java Agent Development environment (JADE).

Multi-Agent systems have been in develop for many years, and researchers have tried to implement them to improve the Smart Homes. In the paper *A Multi-Agent Home Automation System for Power Management* [4], a negotiation protocol is developed to adapt power consumption to the available power resources according to user comfort and cost criteria.

Some other research includes different types of negotiation in the same simulation, as in *A Multi-Agent System Performing One-to-Many Negotiation for Load Balancing of Electricity Use* [3]. In this paper, three different methods of negotiation can be found. The offer method, which is a one-step negotiation where only one bid is placed. The request for bids method where agents make a bid of the amount they want to pay for the electricity, and if an agreement is not reached they make new bids until the agreement is finished. The announce reward tables method

Another example of negotiation can be found in *A Generic Negotiation Model for MAS Using XML* [12]. Here, the negotiation has been built on three levels: communication level, negotiation level and strategic level. The negotiation protocol presented is characterised by messages exchanged between an initiator and participants, as in the Contract Net Protocol framework defined by Reid G. Smith in 1980.

In this project a sum of all the ideas related to this research will be implemented to try to encompass all of this in one Smart House and to take the Smart House environment to a higher complexity level. So that when the basic Smart House designed in the project will be implemented, it will be easy to expand it and to bring it to an actual house.

1.4 RULES AND EXPERT SYSTEM

1.4.1 EXPERT SYSTEMS

An expert system is a computer system that emulates the decision-making ability of a human expert. It usually involves a knowledge base, an inference engine and an interface (Figure 1.2). The expert system implemented will be a weighted fuzzy rule-based system.

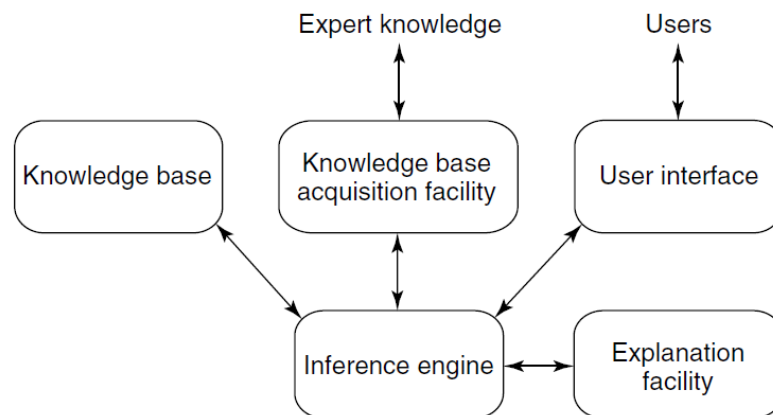


Figure 1.2. Architecture of a simple expert system

The knowledge base will be represented in the form of IF-THEN rules. There are two different kinds of inference engines that can be used: forward chaining and backward chaining. In a forward chaining system, the initial facts are processed and the rules are used to reach a conclusion according to those facts (Figure 1.3). In backward chaining, the system tries to reach the desired conclusion and then finds the conditions that need to be true. In this project forward chaining will be used.

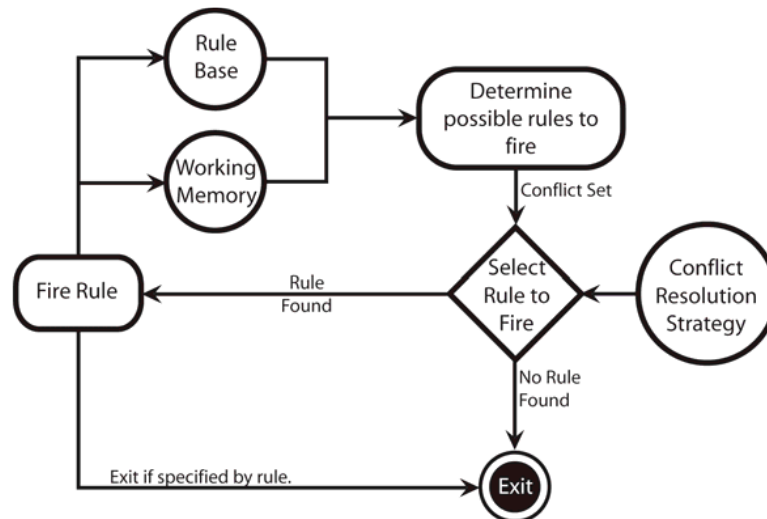


Figure 1.3. Forward chaining system [23]

The inference engine is the software that connects the knowledge base to the user interface. The user interface will be used to display the data and to let the user interact with it. The user can manipulate the data and then the inference engine will find which rule has to fire according to the rules defined in the knowledge base.

1.4.2 RULES

Knowledge, as commonly defined in Artificial Intelligence, will be represented in the form of IF THEN rules with the following form:

Rule: IF Conditions THEN conclusion.

Example 1: IF IsOn(X) THEN TurnOff(X)

This means that if the device X is on then it will be turned off.

Example 2: IF IsOn(X) AND DaylightHours(H) THEN TurnOff(X)

This means that if the device X is on and the actual hour (H) is a daylight hour, the device will be turned off.

There is no ELSE on the rules as all the rules are evaluated. The AND, OR and NOT operators can be used. In logic, a rule is fired when all the conditions in it are true. Some programming languages are built on these concepts such as Prolog, which is the one that will be used. The conclusion of the rules will be turning off or on the device, as that is the range of the project.

To deal with uncertainty and ignorance it is needed to keep track not only of the knowledge, but how sure it is. A scoring system for rules, fuzzy logic for the conditions that have a range and default assumptions to handle the sensors' accuracy will be used to take into account all those factors.

That means that a rule will not be fired when all the conditions are true, it will be fired when its score is the highest score of the rule-set. The score will be given based on: the accuracy of the condition match, the weighting of each condition and the accuracy of the sensor.

1.5 MULTI-AGENT NEGOTIATION

A Multi-Agent system is one that consists of a number of agents that interact with one-another. In the less complex scenarios, all agents are programmed by the same team and they collaborate to complete a task. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate and negotiate with each other, as much as people do.

In systems composed of multiple autonomous agents, negotiation is the key form of interaction that enables groups of agents to arrive at a mutual agreement regarding some belief, goal or plan. The process of negotiation may be of many different forms, such as auctions, protocols in the style of the contract net or argumentation.

1.5.1 NEGOTIATION PARTS

Three different parts must be mentioned regarding Multi-Agent negotiation: the agents, the objects and the protocol [6].

1.5.1.1 AGENTS

The definition according to Wooldridge and Jennings [18] is: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in that environment in order to meet its delegated objectives".

An agent is defined by three different types of behaviour: pro-active, reactive and social ability [16].

Pro-activeness means that reacting to the environment is easy and they are goal driven. They are not driven solely by events, they want to achieve goals. A reactive system is one that keeps the interaction with the environment and responds to the changes that occur in it.

The social ability in agents let them interact using cooperation, coordination and negotiation. Cooperation means that the agents will work together to achieve a common goal. Coordination means managing the interdependency between activities. And negotiation is the ability to reach agreements on matters of common interest.

1.5.1.2 NEGOTIATION OBJECTS

The negotiation objects are the parameters over which the agreement must be reached. It can be a single parameter such as price, or multiple parameters as: price, waiting time, predefined priorities, etc. In the simplest case the agreement is fixed by accepting or rejecting the proposed offer. In more complex cases, the values of the parameters are changed through negotiation and counter-proposals.

1.5.1.3 NEGOTIATION PROTOCOLS

They are the set of rules that govern the interaction. Including the negotiation states (i.e. accepting bids, negotiation closed) or the events that cause transitions (i.e. no more bidders, bid accepted). They also include the role that the agents play on the negotiation (i.e. buyer and seller).

1.6 SIMULATION ENVIROMENT

In the original specification document it was proposed that the simulation would be done using the program *Matlab*. With further research done it was found that the best way to approach the problem was to use a Java interface for the multi-agent system and the user's interaction and the program *Prolog* to define the rule-set.

Also, in that document there was no recollection of a Multi-Agent negotiation system. It was thought that in order to have a more complete and complex simulation it would be interesting to have different agents sharing a common electrical limit. With that on, they would need to interact and negotiate when they reached the electrical limit.

Furthermore, a timer will be added to implement all the desired functions in real time. That means that the set of rules will be fired every second to simulate a real-time simulator. Twenty-four seconds of simulation will be counted as a day, therefore a second will represent a real hour in the house.

1.7 USER INTERFACE

The interface of the program will be developed using Java. It will display all five houses with their respective devices in each house. All houses will have the same devices and they will be represented with a button to switch them on and off and a picture to identify them. There will be text areas to display the history, the power consumption of each house, the total electricity consumption and the multi-agent negotiation: the bids, the demands from each agent and the final agreement with the action that the demanding agent will have to pursue.

It will be an interactive program, so the user will be able to switch on and off devices or start negotiations with the other houses.

In the initial simulation the system displayed an interactive map of the house. When the five agents were added for the Multi-Agent negotiation, the map was deleted. The map required a considerable amount of screen space, and as five houses were controlled there was no need for five interactive maps.

1.8 SYSTEM ESPECIFICATION

The simulation will be developed using the program Netbeans (Java) and the set of rules will be defined in Swi-Prolog. The development will be done in PC using Windows 7 as the operative system.

1.9 EXPECTED OUTCOME

The expert system will manage the devices with the given set of rules and the user will be able to control the status of the devices. The amount of active rules from the rule-set will be chosen by the user running the simulation. The multi-agent negotiation system will allow the agents to interact and achieve a mutual goal while keeping the energy consumption below a certain level.

2. SPECIFICATION

The technologies used to develop the project are listed and explained.

2.1 PLATFORM

The user interface of the program has been created using NetBeans and the programming language used is Java. Java is the best programming language to implement an easy connection between the user interface and the expert rules defined in Prolog.

The Operative System for developing and testing the software has been Windows 7.

2.1.1 NETBEANS

NetBeans [24] is an integrated development environment (IDE) for developing primarily with Java. All the functions of the IDE are provided by modules. Each module provides a well-defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately.

The NetBeans' GUI design-tool enables developers to prototype and design Swing GUIs by dragging and positioning GUI components.

2.2 PROGRAMMING LANGUAGES

2.2.1 PROLOG

Prolog [25] (programming in logic) is a programming language with its roots in mathematical logic. It has a declarative paradigm that makes it apt for a number of areas of Artificial Intelligence, including problem solving and knowledge based systems (expert systems). Being Prolog a widespread programming language it became an ISO standard in 1996, which was based on the Edinburgh syntax, also known as DEC-10 syntax. Prolog has large number of concepts used in a recursive style of programming, such as clauses, facts, rules and procedures.

Prolog has been used to define the rules in the expert system and to find the rule with the highest score, which will be the one that will be fired. The actions that will be done after firing the rule have been also defined in Prolog (The THEN part of an IF-THEN rule).

2.2.2 JAVA

Java [26] is a high-level programming language developed by Sun Microsystems. It is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java interpreter.

Java is popular as its compiled code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows.

Java is a general purpose programming language with a number of features that make the language well suited for use on the internet. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser.

2.3 EXTERNAL SOFTWARE

2.3.1 SWI-PROLOG

SWI-Prolog [27] is an open source implementation of the programming language Prolog. A part of SWI-Prolog is JPL, which is a bidirectional interface between Java and Prolog. It requires the Java Development Kit. SWI-Prolog was used to define the rules of the expert system.

2.3.2 JDK

The Java Development Kit (JDK) [28] is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation. It is used to develop Java applications and applets. It consists of a runtime environment and the tools and programming that developers need to compile, debug, and run applets and applications written in the Java language.

3. DESIGN

3.1 SYSTEM FUNCTIONS

In the system all parts of the simulation will be controlled by the user in charge of it. This is not a real application but a simulation that tries to show how an expert system and a multi-agent negotiation could be implemented in a house in order to convert its status to a Smart House.

The main function of this system is to demonstrate how to implement all the methods listed above. If the application wants to be released to the actual market, changes need to be done to adapt it to the household user.

3.2 USER REQUIREMENTS

As said before, the application will be developed as a one user application. All five houses will be controlled by the same user. That means that instead of stating the user requirements it makes more sense to listen the agent requirements, because those are the ones that will emulate the behaviour of real persons.

In the real application there would be five different applications so that there is one application for each house. Here the agent requirements will be listed and the difference between the real requirement and what the simulation will be able to do will be explained:

- The agent wants to control the devices of his house and to know the status of them at any time. The user in charge of the simulation will change the status of the devices by clicking on the icon and the information log will display all events related to the simulation.
- The agent wants to be able to negotiate with the others so he can turn on a device when the power limit has been reached. The negotiation will start automatically when the user tries to turn on a device that will make the total electrical consumption reach the established limit.
- The agent wants to have a high level of temperature comfort. The user will control it starting the negotiation when he thinks it is necessary.

3.3 INTERFACE DESIGN

The interface has been designed to simulate an application on a tablet. Every function will be done tapping on the screen. In this case, as we are simulation the application in a PC, the user will need to click on the button.

The interface has been created using NetBeans IDE 7.4 and its Swing GUI Designer in Java. There are four different parts that can be distinguished.

The first part in the centre of the application shows the five houses with their devices. The user can click on the device's buttons to turn them off or on. The buttons will change colour according to the status of the device: red when the device is off and green when the device is on. Also, the user will be able to control the level of the fridge, by introducing the amount of food kilograms that there are in the fridge at that moment. In the real simulation the weight of the food in the fridge would be calculated using weight sensors instead of the user having to add the data to the program.

The second part is situated in the right side of the app and displays general information: such as the electrical consumption, the log displaying information of the devices and the hour of the day.

The third part is in the bottom of the app and it displays the one bid negotiation protocol: the information, the request button to start the transaction and the button that allows the demanding agent to accept the bid.

The last part is in another window so that there is enough space in the screen to fit everything. This window will include the heat control system. Displaying the bids of the agents and their satisfaction. There will be a button which the user will press when he wants to start the negotiation.

3.3.1 DEVICES

Each house will be in control of eight devices and they will be the same for all the houses. The characteristics of the devices are displayed in the table below:

Device	Energy consumption (kW)	Brief description
Phone Charger	2	Phone charger situated in the bedroom
Light	60	Roof light situated in the living room.
Computer	50	A desktop computer situated in the studio.
Computer Screen	40	The computer screen of the desktop mentioned above.
Electric car	3000	Electric car being charged in the garage.
Fridge	800	Fridge with freezer situated in the kitchen.
Television	500	Large-screen television situated in the living room.
Washing machine	700	Washing machine situated in the kitchen.

Table 3.1 Description of the devices displayed in the simulation

4. METHODS: RESEARCH METHODOLOGIES

4.1 EXPERT SYSTEM

4.1.1 RULE MATCHING

The system will be able to, given a set of rules, evaluate the current state of the devices against the rules. Every time unit the rule-matching algorithm will be fired. Once all the rules have been evaluated and a score has been given, the algorithm will fire the rule with the highest score, also referred as the “winning rule” or “fired rule”.

When the conditions of a rule imply a range of values, instead of a true or false, a fuzzy matching criteria will be used to evaluate that condition.

4.1.2 FUZZY MATCHING

Fuzzy matching will be used to calculate the score value of a condition from zero to one. Being one the perfect match and zero no match at all.

$$A: X \rightarrow [0,1]$$

For each $x \in X$ the value $A(x)$ expresses the degree of membership of the element x of X in standard fuzzy set A . In the project standard fuzzy set will be used, they state that membership degrees are expressed by real numbers in the unit interval [11].

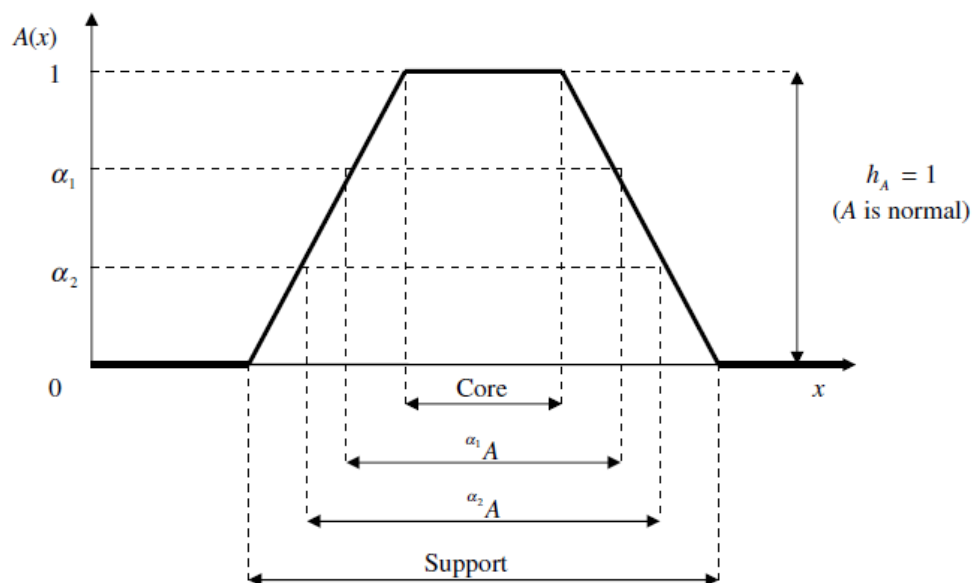


Figure 4.1. Illustration of some basic characteristics of fuzzy sets. [11]

As it can be seen in the Figure 4.1, one of the most important concepts related to standard fuzzy sets are the concepts of an α -cut and a strong α -cut.

Given a fuzzy set A defined on X and a particular number α in the interval $[0, 1]$, the α -cut of A, denoted as A^α , consists of all the number with a membership greater than or equal to α . It can be defined as:

$$A^\alpha = \{ x \mid A(x) \geq \alpha \}$$

The strong α -cut is the same as an α -cut but it only includes all the numbers with a membership greater than α :

$$A^{\alpha+} = \{ x \mid A(x) > \alpha \}$$

Hence:

$$A^{\alpha_1} \subset A^{\alpha_2} \mid \alpha_1 \geq \alpha_2 \qquad A^{\alpha_1+} \subset A^{\alpha_2+} \mid \alpha_1 > \alpha_2$$

Fuzzy sets will be defined for the conditions that require it. In this simulation they will be functions related to the control of time.

4.1.3 WEIGHTING

The conditions in the rules will be given weights according to the importance of each condition. This will be one of the parameters in the total score of each rule.

4.1.4 ACCURACY

The state of the devices in this project will only have two values: on and off. The sensors displayed around the house will be in charge of transferring this information to the computer to process it. There is a loss of information in each transaction and, also, the sensors have an accuracy. The accuracy of a sensor is defined by the Oxford Dictionary as: “The degree to which the result of a measurement, calculation, or specification conforms to the correct value or a standard”.

A random number will be used to model this uncertainty related to the transfer of information. The value of this number will change every time unit and it will be between 0.6 and 1.

4.1.5 SCORE FORMULA

A score is given to each rule in the set. This score is defined as the degree to which the conditions of the rule match the actual status of the devices. The rule with the highest score is the one that will be fired.

To calculate the score two different components will be used: weighting and fuzzy match. The use of weight in the conditions in a rule determines that some conditions are more important than others. The weight is a real number between zero and one. $W = [0, 1]$. The fuzzy match determines the degree of matching of the condition. It is also a real number between zero and one. $F = [0, 1]$.

The score formula will be defined as:

$$S(C_i) = W(C_i) \cdot F(C_i) \quad (4.1.1)$$

$$K(Rule_i) = \sum W(C_i) \quad (4.1.2)$$

$$S(Rule_i) = \frac{\sum S(C_i)}{K(Rule_i)} \quad (4.1.3)$$

The rule that will be fired will be the one with the highest score. To avoid firing rules with really low scores a minimum score will be established for the rule to be fired.

4.2 MULTI-AGENT NEGOTIATION

4.2.1 HEAT CONTROL SYSTEM

4.2.1.1 AGENTS

Five agents will negotiate to try reach a common level of satisfaction. The negotiation will be started by the agent with the lowest level of satisfaction. They will need to find a new situation in which the satisfaction of the demanding agent is higher but without surpassing the electrical limit for the heating system, which is independent from the total electrical consumption.

The satisfaction of the agents, or their human thermal comfort, is a combination of a subjective sensation and several objective measures. That means that there is not a temperature at which everyone is comfortable. To implement that, the five agents will have different comfort functions.

There is a international standard ISO 7730-2005 that provides a method to evaluate comfort, it is known as Predicted Mean Vote (PMV) [2] and it relies in a large number of values to calculate the comfort of the user. As in this project the thermal comfort is a small part of the Smart House Environment we will define the satisfaction function as a lineal function that only depends on the house temperature. For further work, if a better comfort function wants to be implemented more values need to be evaluated and used to calculate the satisfaction measure.

4.2.1.2 NEGOTIATION OBJECTS

Heat: The electrical energy used for heating in the house. This is a number between 1,500 W and 4,000 W [27]. But as the behaviour of the agents is determined by their satisfaction regarding the temperature of the house, the heating energy will be usually between 2,500 W and 3,500 W.

Percentage (P): The percentage of watts that the agent will reduce from his heating system. It depends on the satisfaction of the user, if the agent has a high satisfaction number he will be willing to decrease his heating more than if the agent has a low satisfaction. After the first bid, the percentage of kilowatts that the user will reduce it will be lower.

RandomNumber (R): This will add uncertainty to the formula, as an agent does not behave in the same way in a similar situation twice. This will be a number between zero and two. This uncertain number will be multiplied by a lineal function that depends on the Percentage. This function has a minimum value of 0.2 and a maximum value of 1.2.

WantsToNegotiate (WN): This is a feature that allows the agent to avoid taking part in the negotiation. If the agent does not want to participate in the negotiation, his bids will be always the amount of his heating. It will be zero if the agent does not want to participate and one if he does. For the simulation, it has been stated that five percent of the times the agent does not want to participate.

The First Bid will be calculated using (4.2.1) and the following bids will be calculated using (4.2.2).

$$\text{First Bid} = \text{Heat} \cdot [(1 - P) - (P \cdot 0.42 - 0.2) \cdot R] \cdot WN \quad (4.2.1)$$

$$\text{Next Bid} = \text{Heat} \cdot [(1 - P) - R] \quad (4.2.2)$$

If the bid that has been calculated using one of both functions requires the agent to decrease his satisfaction under forty-five percent, that agent will not be participating in this negotiation anymore. He will stay with the same bid during the entire process.

4.2.1.3 NEGOTIATION PROTOCOL

The negotiation protocol implemented will be a typical bidding system [4], but with a few modifications added. The system will use a reverse auction, which means that the roles of buyer and seller are reversed. Instead of buyers competing to obtain a good from the seller at the lowest price possible, here the buyers become the sellers that try to give a good at the highest price.

There are two objectives that want to be achieved using this negotiation protocol: avoid overpassing the maximum available energy and keep the comfort of all the agents over a certain value.

The negotiation system will be formed by one buyer and four sellers $B = \{1, \dots, 4\}$. The buyer will have a desired value (d_j) and the sellers will place their bids (b_i). The negotiation finishes when the sum of the desired value of the buyer and all the bids of the sellers is below the electrical limit (Equation 4.2.3):

$$d_j + \sum_{i \neq j} b_i \leq \text{Electrical Limit} \quad (4.2.3)$$

Equation 4.2.3 is checked after each negotiation loop. The sellers, according to their actual satisfaction, will lower their bids more or less. Also, the buyer will lower his demand so a new state of comfort is reached quickly [1].

The negation will be started by the agent with the lowest satisfaction level. That agent will become the buyer and the other four agents will become the sellers. The buyer will demand a rise of his heating to a certain number, and the other agents will lower their heating to a certain extent. As stated before, the negotiation will be over when (4.2.3) is true.

4.2.2 ONE-BID SYSTEM

4.2.2.1 AGENTS

Five agents will be displayed in the simulation. The user will be able to control the status of the devices of the five houses. To avoid surpassing the electrical limit that has been set up amongst the five houses the agents will have to negotiate if they want to turn on a device when they are about to surpass the electrical limit.

4.2.2.2 NEGOTIATION OBJECTS

The bidding score will be calculated taking into account four factors and their specific weighting (Equation 4.3.1): Hours prior to the shutdown (Weight = 20%), number of devices (Weight = 5%), neighbourhood where the negotiating agents live (Weight = 10%) and the preferences of the demanding agent (Weight = 65%).

Hours prior to the shutdown (H): Number of units of time that the offering agent will wait before shutting down the device (s). This number will be simulated as a random number between 1 and 3.

Number of devices (ND): Number of the devices that the agent needs to shut down in order to let the demanding agent turn on its device without surpassing the electrical limit. In this simulation this number will be one or two.

Neighbourhood zone that the agent lives in (N): The simulation recreates two neighbourhood zones: three agents live in one zone and two in the other one. This number will be a one if both agents are from the same zone and two if they are from different.

Preferences of the demanding agent (D): Preference of the demand, a number from zero to four. Being zero the most desirable action and four the less desirable one.

$$\text{Total score} = 0.2 \cdot H + 0.05 \cdot ND + 0.1 \cdot N + 0.65 \cdot D \quad (4.3.1)$$

Based on the variables' range of values, the total score calculated using (4.3.1) will be a number between 0.35 and 3.7.

4.2.2.3 NEGOTIATION PROTOCOL

The negotiation protocol that will be implanted in the system will be a “Contract Net” or “Take-It-or-Leave-It” protocol. This method is based on the contracting mechanism used by business men to govern the exchange of goods and services. This is one of the simplest method of Multi-Agent negotiation, but given the nature of the problem it is the solution that makes more sense and it would be the method that real humans would apply to that scenario.

The agent that wants a task solved is called the manager and the agents that might solve the task are called potential contractors. The roles of the agents are not specified in advance as an agent can act as a manager by making task announcements and it can also act as a contractor by responding to task announcements. The contract net offers the advantages of graceful performance degradation. If a contractor is unable to provide a satisfactory solution there will be other contractors that the agent could try to negotiate with.

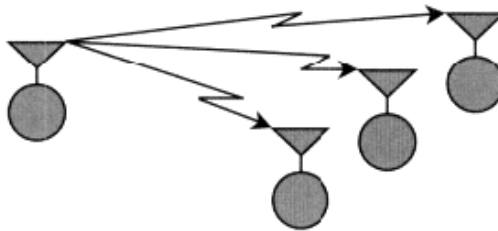
The negotiation process for the manager is:

- Announce the task that needs to be performed
- Receive and evaluate the bids from the potential contractors
- Award a contract to a suitable contractor
- Receive and act according to the results

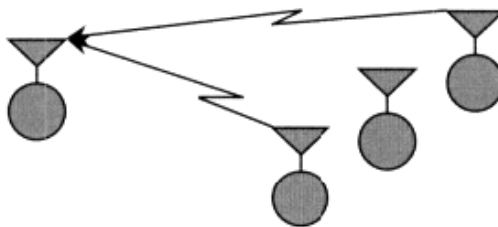
From a contractor's perspective, the process is:

- Receive task announcement
- Evaluate the capability to respond
- Place the bid
- Perform the task if the bid is accepted
- Report the results

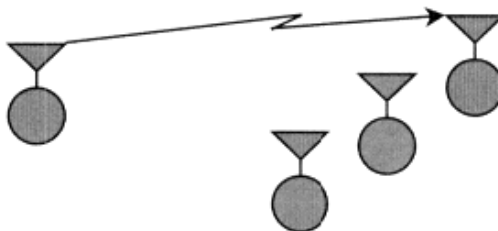
A manager announces the existence of tasks via a (possibly selective) multicast



Agents evaluate the announcement. Some of these agents submit bids



The manager awards a contract to the most appropriate agent



The manager and contractor communicate privately as necessary

Figure 4.2. Basic steps in the contract net

The manager announces the task that wants to perform, the potential contractors evaluate the task (based on the negotiation criteria explained in the section 4.2.2.2) and offers the bid to the corresponding manager. The contractors are not forced to bid, that means that not all of the contractors will be negotiating with the manager.

The manager evaluates all the bids received and chooses the one that suits better the needs that need to be fulfilled. The manager notifies the contractor of the bid acceptance with an announced award message. After the contractor is notified, both parts perform the task that they need to do according to the contract.

In the simulation there will be one manager and four contractors, a total of five agents. The procedure follows as: the manager will be the agent that wants to turn on a device but the electrical limit will be surpassed if he does so. The contractors will be the other four agents, they will offer to turn off one of their devices but they will ask for some action in exchange for it. After the contractors make their bid, the manager agent will select the bid with the lowest score.

Once the contract is settled, the manager will be able to turn on his device but the contractor will have to turn off his. In exchange for that the manager will do one of the tasks explained in the section 5.3.

5. IMPLEMENTATION

5.1 EXPERT SYSTEM

The implementation of the expert system will be done in Prolog [7]. The rule-set will be created using Prolog and then the obtained results will be sent to NetBeans.

5.1.1 RULES

Here all the rules defined to control the expert system will be explained. The score of the rules is calculated multiplying each condition by its weight and the accuracy number. The minimum score for the rules to be fired is 0.6 for the first rule and 0.3 for all the others.

RULE 1

This rule aims to control that the house does not surpass the electric limit. The score takes into account the status of the main devices in the house. There are six conditions in this rule.

IF IsOn(Car) ^ IsOn(WashingMachine) ^ IsOn(TV) ^ IsOn(PC) ^ IsOn(Screen) ^ IsOn(Light)
THEN TurnOff(Car)

- Condition 1: Weighting = 0.9.
- Condition 2: Weighting = 0.7.
- Condition 3: Weighting = 0.6.
- Condition 4: Weighting = 0.2.
- Condition 5: Weighting = 0.2.
- Condition 6: Weighting = 0.3.

The value of the conditions is one or zero. There is no range of values as they state the status of the devices. From now on this type of conditions will be referred as status condition.

rule1(RULE1,CAR,RCAR,PC,RPC,SC,RSC,L,RL,TV,RTV,W,RW,S1):-
*S1 is (0.9*CAR*RCAR+0.7*W*RW+0.6*TV*RTV+0.2*PC*RPC+0.2*SC*RSC+*
*0.3*L*RL)*(1/2.9)*RULE1.*

Example of the notation used for the variables in Prolog. The same notation is used for all the rules in the expert system:

CAR: 1 if car is on, zero if car is off.

RCAR: Accuracy of the sensor that measures the state of the car

RULE1: This is zero if the activation button of this rule is not pressed in the simulation.

RULE 2

This rule controls the turning off of the light during the sunlight hours. There are two conditions.

IF IsOn(Light) ^ SunlightHours(H) THEN TurnOff(Light)

Condition 1: Weight = 0.7.

Condition 2: Weight = 0.9. Fuzzy Match.

```
rule2(RULE2,L,RL,T,S2):-
    calctime(T,TL),
    S2 is (0.6*RL+0.9*TL)*L*(1/1.5)*RULE2.
```

The first condition is a status variable. The second condition is a fuzzy match defined by the equations in Figure 5.1.1.

$$A(x) \begin{cases} x < 6 \rightarrow A(x) = 0 \\ 6 \leq x < 9 \rightarrow A(x) = \frac{x-6}{3} \\ 9 \leq x < 16 \rightarrow A(x) = 1 \\ 16 \leq x < 19 \rightarrow A(x) = \frac{19-x}{3} \\ x \geq 19 \rightarrow A(x) = 0 \end{cases} \quad (5.1.1)$$



Figure 5.1.1. Graph displaying the fuzzy function of the Rule 2

The α -parameters of this fuzzy set are:

$$\alpha_1 = \frac{2}{3} \quad \alpha_2 = \frac{1}{3}$$

RULE 3

This rule turns the computer screen off if the computer is off.

IF IsOff(Computer) ^ IsOn(Screen) THEN TurnOff(Screen)

Condition 1: Weight = 0.9.

Condition 2: Weight = 0.9.

Condition 3: Weight = 0.5. Fuzzy Match.

The first two conditions are state value conditions. The third condition is a fuzzy match to calculate the night hours when there is a higher chance that the user is not using the computer.

```
rule3(RULE3,PC,RPC,SC,RSC,T,S3):-
  calctimeS(T,TS),
  S3 is (0.9*RPC+0.8*RSC+0.5*TS)*(1/2.2)*(1-PC)*SC*RULE3.
```

$$A(x) \begin{cases} 10 \leq x \leq 23 \rightarrow A(x) = 0 \\ 0 \leq x < 3 \rightarrow A(x) = \frac{x}{3} \\ 3 \leq x < 7 \rightarrow A(x) = 1 \\ 7 \leq x < 10 \rightarrow A(x) = \frac{10-x}{3} \end{cases} \quad (5.1.2)$$

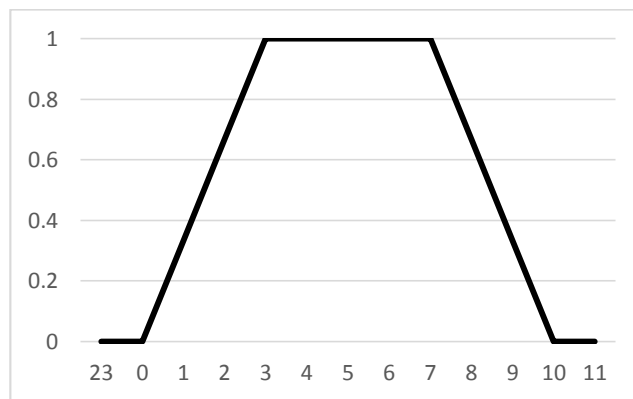


Figure 5.1.2. Graph displaying the fuzzy function of the Rule 3

The α -parameters of this fuzzy set are:

$$\alpha_1 = \frac{2}{3} \quad \alpha_2 = \frac{1}{3}$$

RULE 4

This rule calculates the level of the fridge using the weight provided by the sensor.

IF IsOn(Fridge) THEN CalculateLevel(Fridge)

Condition 1: Weight = 0.9.

Condition 2: Weight = 0.3.

The first condition is a state value condition. The second one gives a number associated with the difference between the actual level of the fridge and the appropriate level that the *calcweight* function calculates. If the difference between the levels is zero the score of this rule will be zero.

```
rule4(RULE4,F,RF,WF,LF,LF1,S4):-
    calcweight(WF,LF1,F),
    S4 is (0.9*F*RF+(LF1-LF)*0.3)*(1/1.2)*F*RULE4.
```

The function *calcweight* calculates the appropriate level of the fridge with the following function (5.1.3).

$$calcweight(WF,LF,F) \begin{cases} WF < 6 \rightarrow LF = 1 \\ 6 \leq WF < 9 \rightarrow LF = 2 \\ 9 \leq WF < 13 \rightarrow LF = 3 \\ 13 \leq WF < 21 \rightarrow LF = 4 \\ WF \geq 21 \rightarrow LF = 5 \end{cases} \quad (5.1.3)$$

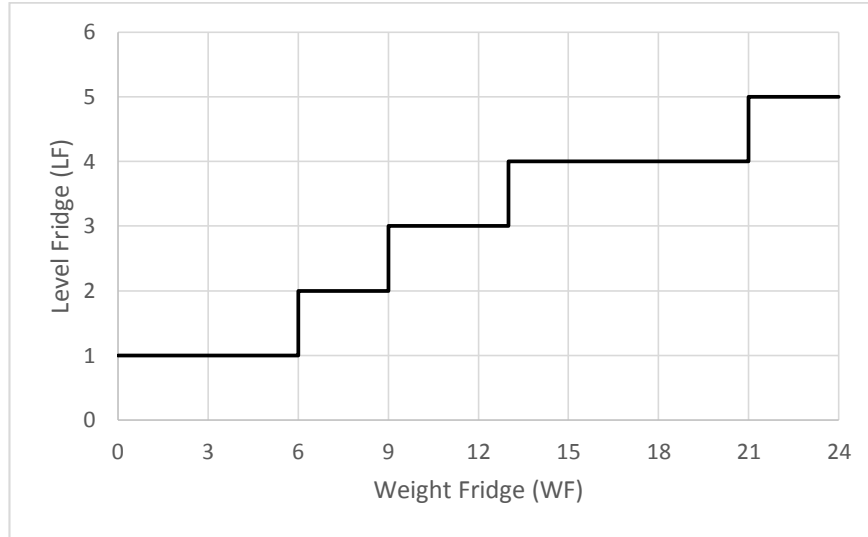


Figure 5.1.3 Graph displaying the calcweight function

RULE 5

This rule turns the light off if the user is watching a movie during night hours.

IF IsOff(Computer) ^ IsOn(Screen) THEN TurnOff(Screen)

Condition 1: Weight = 0.9.

Condition 2: Weight = 0.8.

Condition 3: Weight = 0.7. Fuzzy Match.

The first two conditions are state value conditions. The third condition is a fuzzy match to calculate how close the actual time to the night hours is.

```
rule5(RULE5,TV,RTV,L,RL,T,S5):-
    calctvtime(T,TTV),
    S5 is (0.9*RTV+0.8*RL+0.7*TTV)*TV*L*(1/2.4)*RULE5.
```

$$A(x) \begin{cases} 3 \leq x < 20 \rightarrow A(x) = 0 \\ 20 \leq x < 22 \rightarrow A(x) = \frac{x-20}{2} \\ 22 \leq x < 1 \rightarrow A(x) = 1 \\ 1 \leq x < 3 \rightarrow A(x) = \frac{3-x}{2} \end{cases} \quad (5.1.4)$$

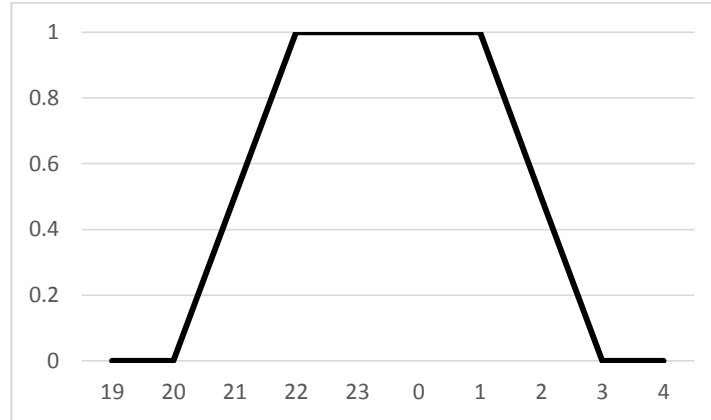


Figure 5.1.4. Graph displaying the fuzzy function of the Rule 2

The α -parameter of this fuzzy set is:

$$\alpha_1 = \frac{1}{2}$$

RULE 6

This rule turns the washing machine off during night hours.

IF IsOn(WashingMachine) ^ NightHours(Time) THEN TurnOff(WashingMachine)

Condition 1: Weight = 0.9.

Condition 2: Weight = 0.9.

The two conditions are state value conditions. The second condition is related to time, but a fuzzy match will not be used as this rule implies a scheduled function so the time has to match exactly what has been defined by the user.

```
rule6(RULE6,W,RW,T,S6):-
    calcwtime(T,TW),
    S6 is (0.9*TW+0.9*RW*W)*RULE6*TW*(1/1.8).
```

RULE 7

This rule turns the phone charger on during night hours.

IF IsOff(Charger) ^ NightHours(Time) THEN TurnOn(Charger)

Condition 1: Weight = 0.9.

Condition 2: Weight = 0.9.

The two conditions are state value conditions and the second condition is not a fuzzy match for the same reasons as the previous rule.

```
rule7(RULE7,C,RC,T,S7) :-  
    calcctime(T,TC),  
    S7 is (0.9*TC+0.9*(1-C)*RC)*RULE7*(1/1.8).
```

5.1.2 EXPERT SYSTEM CODE

The system has a timer and each second represents an hour in the simulation. Every second the five jStep buttons are pressed automatically, allowing the expert system to find the best status of the devices according to the rules defined in Prolog.

The status of the devices are saved in Strings. If the variable value is one that means that the device is on and if it zero that it is off.

jStepActionPerformed: It calculates a random number for each device, simulating the accuracy of the sensors as previously cited. It also calculates and displays the electrical consumption of that agent (*CalculateEnergy*) and the sum of the five agent's electrical consumption (*CalculateTotal*).

It sends all the variables to Prolog and then Prolog returns the next status of the devices. Then, in NetBeans again, it changes the status of the devices according to the values provided by Prolog.

5.2 HEAT CONTROL

5.2.1 PHASES OF THE NEGOTIATION PROTOCOL

There are three phases that can be differentiated: the energy demand, the negotiation and the final decision.

The energy demand phase starts with the calculation of the agent with the lowest satisfaction level and his desired heat. To calculate the satisfaction level a pre-defined function will be used.

5.2.1.1 THE SATISFACTION FUNCTION

The agents have been organized regarding their heating preferences: the fifth agent is the easiest to please and the first agent is the hardest as he requires the highest heating level. The satisfaction function is a linear function defined as (5.2.1).

$$Satisfaction = \frac{Heat - (2000 - Agent * 100)}{20} \quad (5.2.1)$$

In the figure below the different satisfaction functions between the agents are displayed.

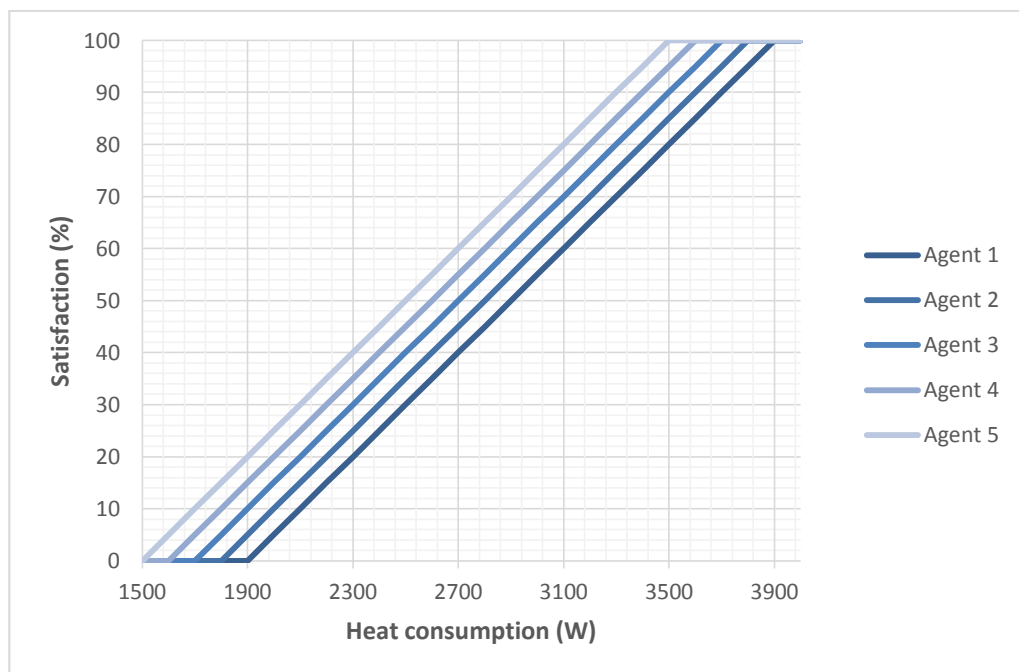


Figure 5.2.1 Graph displaying satisfaction level versus heat consumption for each agent

After this initial phase the negotiation starts. The agents place their bid and the negotiation algorithm checks if an agreement has been reached, the electrical limit for the heat is 14000W. If an agreement has not been reached the next round of bids starts and also the demanding agent will lower his demand. That will help to find a solution quickly. If after six bids the agreement has not been reached, the negotiation is over. That is the last phase of the process, the final decision.

5.2.2 NEGOTIATION 2

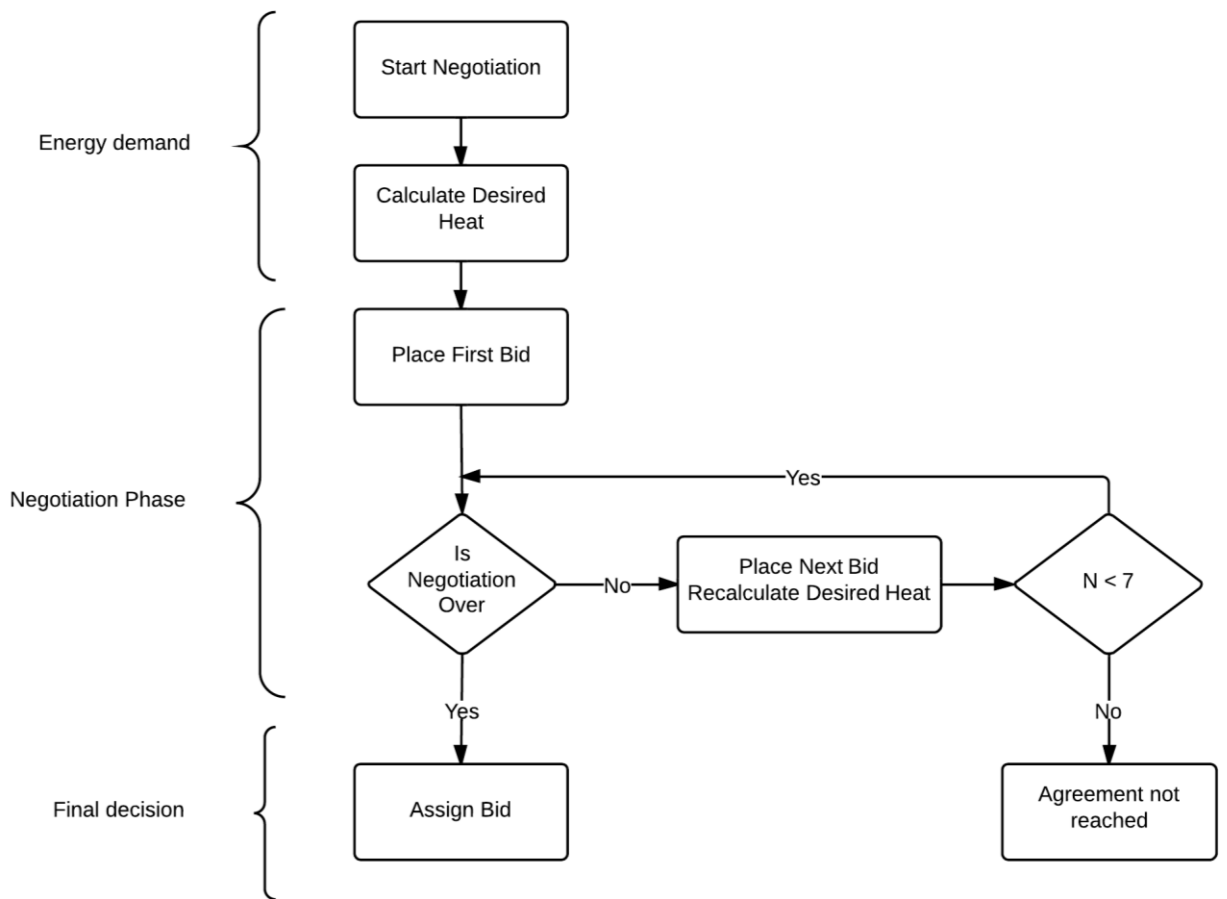


Figure 5.2. Flux chart describing the negotiation process

The negotiation process starts by looking for the agent with the lowest satisfaction (*StartNegotiation*).

LowestSatisfaction: This function finds the agent with the lowest satisfaction level. That will be the agent that will start the negotiation. To calculate the satisfaction level of each agent the *Satisfaction* function will be used.

DesiredHeat: This function, given an agent and its satisfaction returns the desired heating that the agent wants to implement.

NegotiationPhase: This is the main function of the negotiation process. The agents place their bids until the agreement is reached. There is a limit of 6 bids, if the agreement is not reached a new negotiation will be started from scratch.

At the beginning of each loop in the negotiation phase, it is checked if the negotiation is over and if it is not the offering agents place the next bids, which are calculated using the function described in section 4.2.2.2. Inside them there are four other functions that will be used: *FOffer*, *NOffer*, *WantsToNegotiate* and *ContinueBidding*.

FOffer: This function, given the satisfaction level, returns the percentage of watts that the agent will reduce in his bid. This is only used for the first bid of the negotiation.

NOffer: This function does the same thing as the one explained before but it is used for the second bids, third bids and so on. The difference between the function is that the percentage is lower than in the initial bid.

WantsToNegotiate: This returns a zero ten percent of the times and one the other ninety percent of the times. It represents the probability of the agent not being interested in taking part in the negotiation process.

ContinueBidding: If an agent has to place a bid that will lower his satisfaction level under fifty percent, he will not place that bid and that agent will stop negotiating with the others.

IsNegotiationOver: This is a function that returns a Boolean value that is true if the sum of all heating values are under the heating limit.

AssignBid: This function assigns the final bids to the actual values of the agents. It will only do that if the negotiation has been successful.

5.3 CONTROL NET NEGOTIATION

5.3.1 PHASES OF THE CONTROL NET

As said before, there is no negotiation phase in this one bid system. Therefore, there will be two phases: the device demand phase and the decision phase. The limit of total electrical consumption is 13000W.

In the device demand phase the demanding agent starts the negotiation by stating what device does he want to turn on. Next, the other agents will state their demands following their preferences. To simplify the behaviour of the agents, each one has always the same request.

- Agent 1 asks to the demanding agent to cut the grass in his house.
- Agent 2 asks to the demanding agent to cook during one day for her.
- Agent 3 asks to the demanding agent to clean his house during 2 hours.
- Agent 4 asks to the demanding agent to give her money.
- Agent 5 asks to the demanding agent to do his laundry.

In the decision phase the demanding agent will choose the bid with the lowest score. The score has been explained previously but there is an important part of the score: the preference of the deciding agent. Each agent has different levels of preference towards the five different offers that can appear in the simulation.

The agent's preferences are listed in the table below. Each action has a score from zero to four. Being zero the most desirable and four the less desirable.

Score	0	1	2	3	4
Agent 1	MONEY	CLEAN	LAUNDRY	COOK	CUT
Agent 2	CLEAN	LAUNDRY	CUT	MONEY	COOK
Agent 3	LAUNDRY	MONEY	COOK	CUT	CLEAN
Agent 4	CUT	LAUNDRY	CLEAN	COOK	MONEY
Agent 5	COOK	CLEAN	MONEY	CUT	LAUNDRY

Table 5.3 Preferences of the agents

To implement it in java, a matrix with numbers from 1 to 5 will be created. Each number will correspond to the demand of that agent. For example number 4 is the demand of the agent 4: to give her money.

4	3	5	2	1
3	5	1	4	2
5	4	2	1	3
1	5	3	2	4
2	3	4	1	5

Table 5.4 Numeric approach to the agents' preferences

5.3.2 NEGOTIATION 1

The function that starts the negotiation will be the function “*Total*”, which is fired every time the user presses a button to switch the status of a device.

Total: If the device is on then it will shut down immediately. However, if the device is off and the user wants to turn it on the algorithm will check if the consumption of the device will surpass the electric limit. If it does not surpass the limit the device will be switched on, but if it surpasses the limit it will not allow to switch on.

To start the negotiation the user has to click on the request button and after click on the device that he wants to turn on. The negotiation will take place in the function called “*Negotiation*”.

Negotiation: This function returns one if the negotiation has been successful and zero in the other case. Inside this function there are five other functions that develop the process of negotiation: *Devices*, *FindNumberDevices*, *BidsA*, *BestDevice*, *ActionBid*.

Devices: Given the amount of electricity necessary to switch on the device it calculates the devices that each agents needs to turn off to allow the demanding agent turn on his device without surpassing the limit. It finds the device with the minimum amount of electricity that can be switched off to satisfy the demand. It can be one device or the combination of two devices.

FindNumberDevices: Given the device that has to be switched on by each agent, the function returns an integer with the number of devices that have to be switched off.

BidsA: This function returns the number of agent that places the lowest bid, also known as the winning agent. Inside this function there are two other functions: *OneBid* and *BestBid*.

OneBid is used to place the bid of each agent taking into the account the algorithm previously explained. The most important parameter of the Bid is the preferences of the agent and it will be calculated using the function “*ScoreDesire*”.

BestBid finds the lowest value between the four bids. *BestBid* needs the request button of the offering agent to be pressed in order to find the lowest value, if it is not pressed it will return a zero.

BestDevice: This function returns a string with the device that the winning agent needs to put off.

ActionBid: This function takes into account all the results from the previous functions and it switches off the device of the winning agent and it turns on the device that the demanding agent wanted to use.

6. RESULTS AND ANALYSIS

6.1 ANALYSIS OF THE RULE BASED SYSTEM

In this section, the rule-based system will be analysed by studying the results given certain initial conditions. Only one agent will be analysed as each agent has his own expert system and all of them have the same rules.

In each analysis, the initial situation will be defined by stating the devices that are on and the hour of the day. A table with the score of the rules at that time will be displayed to see the score of the rule that has been fired. Finally, the simulation will continue and if more rules will be fired they will be written down.

6.1.1 SITUATION 1

- Initial devices: Light
Car
Fridge
Washing Machine
- Hour of the day: 10
- Rule fired: Rule 2: Turn off the light

Simulation	Score						
	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
1	0.48726	0.99173	0	0.62603	0	0	0.32345
2	0.52633	0.96917	0	0.72315	0	0	0.41430
3	0.51839	0.86562	0	0.55965	0	0	0.43500
4	0.51210	0.99148	0	0.74933	0	0	0.32495
5	0.61648	0.95472	0	0.65138	0	0	0.44160

Table 6.1.1. Five simulations to compare the same situation

Hour	Rule	Consequence
10	Rule 2	Turn off the light
11	Rule 4	Check the level of the fridge
23	Rule 7	Turn on the phone charger
0	Rule6	Turn off the washing machine

Table 6.1.2 Rules that will be fired in this analysis

One rule is fired on the 10th hour, and after this rule three more will be fired. The fourth rule will always be fired if the fridge is on and the other rules are not fired. That is why the rule is fired on the 11th hour, the same scenario will happen in the third situation (Section 6.1.3).

6.1.2 SITUATION 2

- Devices that are on: Phone charger
Screen
Car
TV
- Hour of the day: 22
- Rule fired: Rule 3: Turn off the computer screen

Simulation	Score						
	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
1	0.40765	0	0.65049	0	0	0	0
2	0.41433	0	0.72519	0	0	0	0
3	0.46071	0	0.72347	0	0	0	0
4	0.40907	0	0.78176	0	0	0	0
5	0.48941	0	0.68139	0	0	0	0

Table 6.1.3. Five simulations to compare the same situation

Only the third rule will be fired in this analysis

6.1.3 SITUATION 3

- Initial devices: Light
Pc
Car
Fridge
Washing Machine
- Hour of the day: 6
- Rule fired: Rule 7: Turn on phone charger / Rule 6: Turn off washing machine

Simulation	Score						
	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
1	0.633131	0.33736	0	0.749275	0	0.87115	0.87355
2	0.5329	0.28144	0	0.56595	0	0.89665	0.9494
3	0.570245	0.56152	0	0.497775	0	0.84305	0.3159
4	0.633583	0.39964	0	0.560475	0	0.96665	0.94215
5	0.497766	0.2838	0	0.57285	0	0.8615	0.9404

Table 6.1.4. Five simulations to compare the same situation

Hour	Rule	Consequence
6	Rule 6	Turn off the washing machine
7	Rule 2	Turn on the light
8	Rule 4	Check the level of the fridge
0	Rule 7	Turn on the phone charger

Table 6.1.5. Rules that will be fired in this analysis

In this analysis two rules can be fired on the 6th hour, depending on the randomness of the moment, one or the other will be fired first. After these two rules, two more rules will be fired.

6.1.4 ANALYSIS IN FURTHER WORK OF THE RULE-BASED SYSTEM

The analysis of the rule-based system does not give a lot of information regarding the system as the environment is considerably simple. The devices only have two values, on and off, that implies that the rules can only draw two different kind of conclusions: turn on or off the device.

With the extension of the simulation, more devices will be added and then more rules with a higher level of complexity, a better analysis of the system could be done. The addition of devices with more than two states would also provide a scenario to develop complex rules.

6.2 ANALYSIS OF THE HEAT CONTROL NEGOTIATION

6.2.1 BIDS ANALYSIS

The evolution of the total energy consumption throughout one hundred simulations is displayed in the Figure 6.3.1.

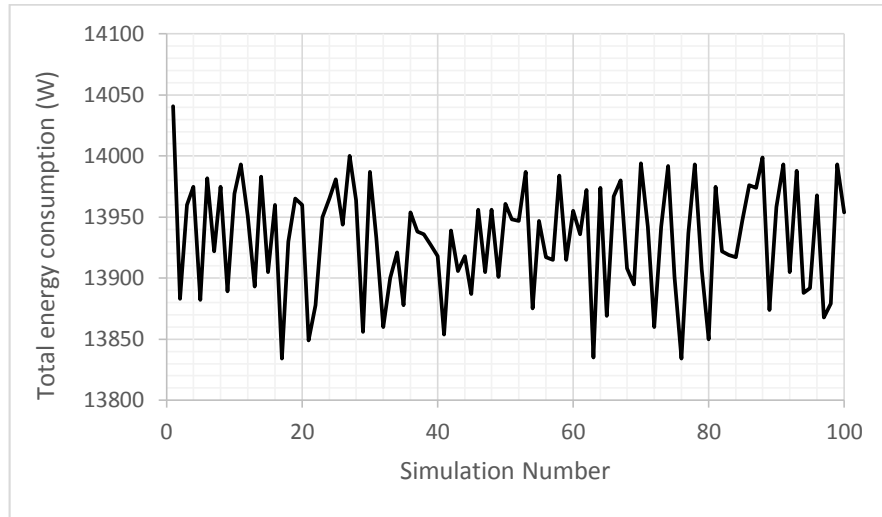


Figure 6.2.1. Graph displaying the total energy consumption.

The system starts with an unbalanced state. The agents are surpassing the energy limit. In the following simulations the total energy consumption oscillates between a minimum of 13834 W and a maximum of 14000 W.

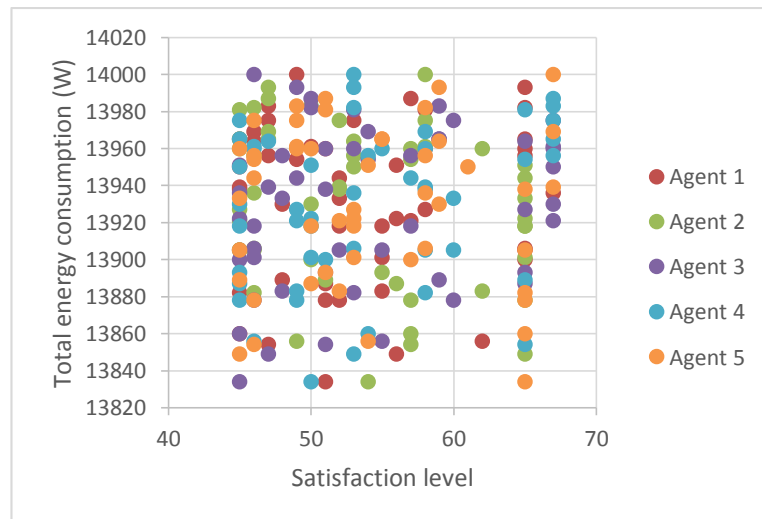


Figure 6.2.2. Graph displaying the relation between total energy consumption and each agent.

There is no apparent relation between the total energy consumption and the different agents, as it can be seen in the Figure 6.3.2. That chart has been done using fifty simulations.

The minimum satisfaction level is forty-five, there is no lower values as this has been establish as the lowest satisfaction under which the agents do not bid. The maximum level of satisfaction during the simulation is sixty-seven.

Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
53.87	53.88	54.02	54.07	53.77

Table 6.2.1 Average value of the satisfaction levels of each agent

In the previous table the average value of the satisfaction level of each agent is shown. This has been calculated using a simulation of one hundred iterations. The average score of all of them is 53.92.

6.2.2 SATISFACTION ANALYSIS

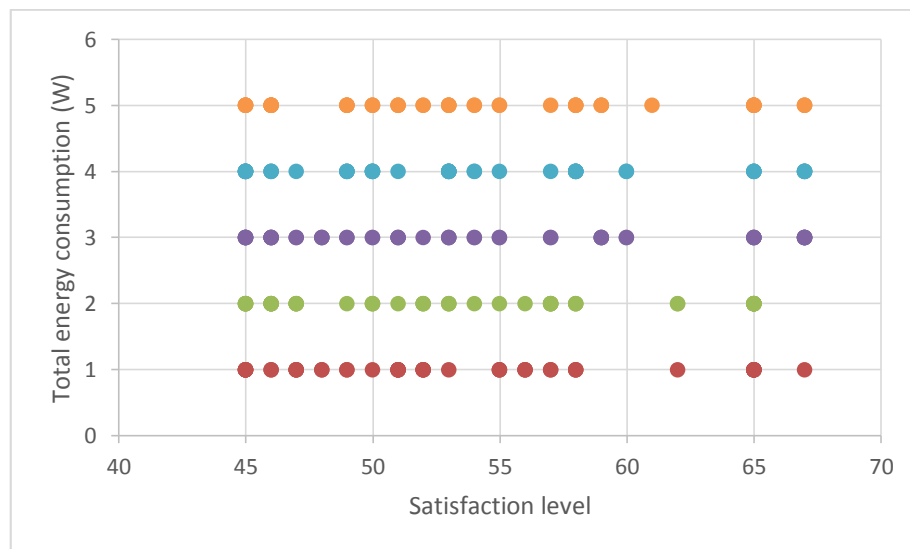


Figure 6.2.3. Graph displaying the different values of satisfaction divided by agent

Fifteen simulations have been made to analyse the agents' satisfaction pattern. In the Figure 6.3.2 the evolution of the agent's satisfaction can be seen. The agents' behaviour follows a cyclic pattern, their satisfaction is encompassed between forty-five and seventy.

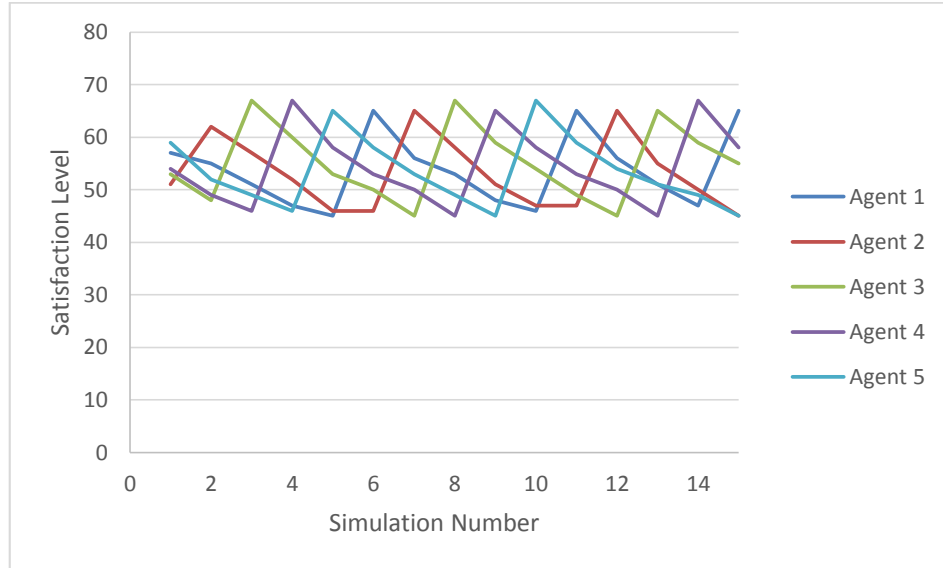


Figure 6.2.4. Graph displaying the evolution of the agent's Satisfaction.

6.3 ANALYSIS OF CONTROL NET PROTOCOL

A Simulation with one hundred bids has been done. Each negotiation has been done with different initial conditions, trying to cover all the different scenarios that could happen in a real negotiation.

6.3.1 INDEPENDENCE BETWEEN AGENTS AND BIDS

In this section the independence between the variables: the agent and the bid placed by the agent will be analysed.

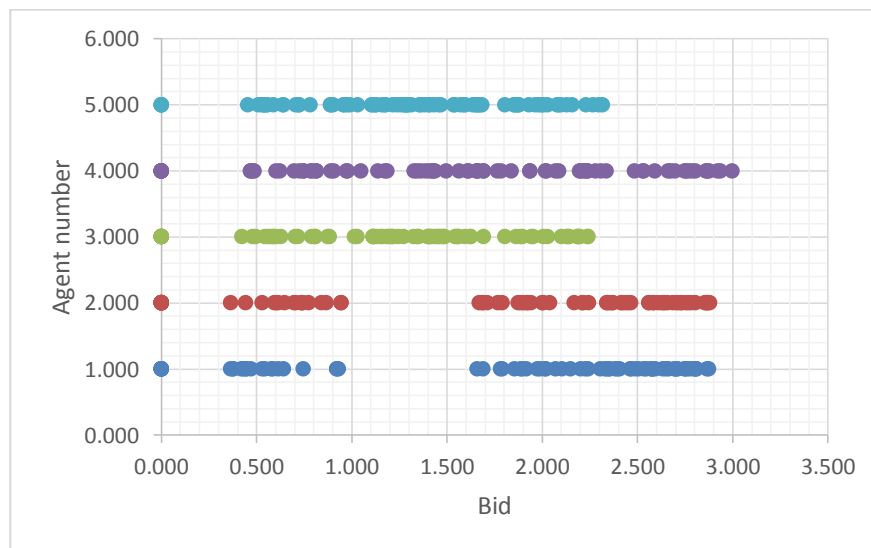


Figure 6.3.1 Graph displaying the relation between the placed bid and each agent.

As it can be seen in the Figure 6.3.1, apparently there is no relation between the bid and the agent that is placing it.

An ANOVA table is created to analyse if the bids of the agents are different from each other.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
Agent 1	79	139.286	1.763114	0.9639
Agent 2	82	134.816	1.644098	1.054559
Agent 3	80	98.092	1.22615	0.413891
Agent 4	83	130.426	1.571398	0.794227
Agent 5	82	107.204	1.307366	0.343726

Table 6.3.1 Summary of the ANOVA Table

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	16.63553	4	4.158882	5.825675	0.000145	2.394193
Within Groups	286.2692	401	0.713888			
Total	302.9048	405				

Table 6.3.2 ANOVA Table

As it can be seen in the table, the F-Snedecor value is more than the limit value so the P-Value is really small. That means that the values are different from each other with a confidence higher than 99.98%.

Some of the agents tend to bid higher and some others tend to bid lower. That will affect the times each agent takes part in the negotiation. As the demanding agent looks for the lower bid and if an agent makes high bids it will not be chosen for the negotiation.

6.3.2 IMPORTANCE OF THE PREFERENCES OF THE AGENTS

The agents' preferences have an important influence in the score of the bid. After all the simulations more than 90% of the negotiations are made with demanding agent's first preference. And the other 10% are made with the demanding agent's second preference

Demanding Agent	Offering Agent	Times
1	3	3
1	4	19
2	1	1
2	3	17
2	5	1
3	5	22
4	1	18
5	2	17
5	3	2

Table 6.3.3. Relation between demanding agent and offering agent

This high dependence of the score of the bid with the agent's preference is due to the fact that the agent's preference is worth 65% of the bid's score.

7. CONCLUSIONS

The development of a Smart House Environment including five Smart Houses has been done successfully. The houses are controlled by a rule-based system and they have to communicate, using different negotiation algorithms, to reach a common goal.

The initial objectives did not include the development of a Smart Home Environment, but only the Simulation of a Smart Home controlled with Intelligent Plugs represented by a Rule-Based Expert System. After this simulation was done, it was decided to expand it adding more intelligent agents creating a small neighbourhood of five agents.

The addition of these agents allowed the development of several negotiation strategies to permit a better control of the comfort of the agents. One of the optional features described in the specification document was the implementation of a temperature control of the house. To implement that, a negotiation protocol was added to the simulation. This feature was implemented using a negotiation protocol that allows the agents to share their energy, while trying to keep a high comfort level without surpassing the energy consumption limit.

The second negotiation protocol is a control net negotiation system. It allows the agents to negotiate to turning on and off of devices. If an agent cannot turn on a device, because it would surpass the electrical limit, the agent can ask the others if they are willing to turn off one of their devices in exchange of house chores.

The further work in this project would be the enlargement of the system and the implementation in real life, instead of only simulating it. Given the original objectives, this project has been extended adding more features than the ones that were thought in the first place. With the development of the project, it has been seen that artificial intelligence can be implemented in multiple aspects of our ordinary life, leading to a total control, in this case, of the user's home.

8. FURTHER WORK

Human knowledge is often inexact and incomplete. It takes a long time for a human apprentice in a trade to become an expert as we acquire knowledge step-by-step. Expert Systems tend to evolve through time as the creator learns more about the system and finds more solutions to the initial problem. That means that there is always work to be done with an expert system. This is one of the main reasons why further work is an important part of the project. This study case has only settled some basic aspects of rule-based systems and the implementation of two negotiation protocols to enhance the user's comfort.

The basic features of the simulation allow to easily expand all the aspects related to it. If all the devices in the house are added to the simulation, then more rules could be implemented and the user would gain a higher control of the status of the applications. In order to apply that, the user would need to add a large number of sensors to his house. But that would allow him to use even more complex rules to control the status of the devices.

In the actual simulation, five agents negotiate between each other to reach a higher comfort level than if they were only by themselves. More negotiation protocols could be implemented such as a water negotiation protocol to try to re-use water, or a common grocery shopping system that would allow the users to have more time for themselves and it would bring savings from the big shopping cart and the supermarket offers.

After all those implementations are done, it would be time to bring the system to a real house and a neighbourhood. That would require a considerable amount of money from the investors, but it would be an excellent opportunity to experience the results in a real simulation. Then, the system could be improved with the experiences learnt from the real environment and its constraints.

9. REFERENCES

9.1 BOOKS AND RESEARCH PAPERS

- [1] S. Abras, S. Ploix, S. Pesty, M. Jacomino. 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008. ICTTA, 2008, pp 1-6.
- [2] V. Bradshaw. The Building Environment: Active and Passive Control Systems, Wiley, 2006, chapter 1.
- [3] F. M.T. Brazier, F. Cornelissen, R. Gustavsson, C. M. Jonker, O. Lindeberg, B. Polak, J. Treur. A Multi-Agent System Performing One-to-Many Negotiation for Load Balancing of Electricity Use, Electornic Commerce Research and Applications, 2002, pp 208-224.
- [4] J. Andrade Cetto, J.-L. Ferrier, J.M.C.D. Pereira, J. Filipe. Selected Papers from the International Conference on Informatics in Control Automation and Robotics 2006, Springer, 2008, pp 59-69.
- [5] J. Choi, E. Lynn Ulserly. System Integration of GIS and a Rule-Based Expert System for Urban Mapping, Photogrammetric Engineering & Remote Sensing, February 2004, pp 217-224.
- [6] D. J. Cook. Multi-agent smart environments, Journal of Ambient Intelligence and Smart Environments 1, 2009, pp 47-51.
- [7] D. Crookes. Introduction to programming in Prolog, Prentice Hall, 1988.
- [8] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents, Int. J. of Robotics and Autonomous Systems 24, 1998, pp 159-182.
- [9] T. Ganu, D.P. Seetharam, V. Arya, R. Kunnath, J. Hazra, S.A. Husain, L.C. De Silva, S. Kalyanaraman. nPlug: A Smart Plug for Alleviating Peak Loads. Third International Conference on Future Energy Systems, e-Energy. May 2012
- [10] B. Huval. Optimization of a Heating System Using Smart Meters & Dynamic Pricing. 2012
- [11] G. J. Klir. Uncertainty and Information: Foundations of Generalized Information Theory,

Prentice Hall International, 1988, chapter 7

[12] P. Mathieu, M.-H. Verrons. A Generic Negotiation Model for MAS using XML, IEEE International Conference on Systems, Man and Cybernetics, 2003, pp. 4262-4267.

[13] M. Pipattanasomporn, M. Kuzlu, S. Rahmanl. An Algorithm for Intelligent Home Energy Management and Demand Response Analysis. IEEE Transactions on Smart Grid, Volume 3 Issue 4, 2012.

[14] M. Sasikumar, S. Ramani, S. Muthu Raman, K.S.R. Anjaneyulu, R. Chandrasekar. A Practical Introduction to Rule Based Expert Systems, Narosa Publishing House, 2007.

[15] Q. Sun, W. Yu, N. Kochurov, Q. Hao, F. Hu. A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation, Journal of Sensor and Actuator Network, 2013, pp 557-588.

[16] N. Vlassis. A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence, Morgan & Claypool, 2007.

[17] G. Weiss. Multiagent systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, 1999.

[18] M. Wooldridge. An Introduction to MultiAgent Systems, Wiley, 2009.

9.2 INTERNET SOURCES

[19] Energy Facts. www.energy-facts.org

[20] World Energy Council. www.wec-indicators.enerdata.eu/world.php

[21] *DUKES – Digest of United Kingdom Energy Statistics*. Annual electricity demand for 2009 for residential, commercial (services) and industrial consumers. 2009

[22] Energy Saving Trust. www.energysavingtrust.org.uk/Electricity/Products-and-appliances

- [23] James Freeman-Hargis, Rule based systems and identification trees. www.ai-depot.com/Tutorial/RuleBased.html
- [23] Edimax. SP-1101W: Smart Plug Switch, Intelligent Home Control. www.edimax.com
- [24] Netbeans IDE. www.netbeans.org
- [25] Swi-Prolog. www.swi-prolog.org
- [26] Java Development Kit (JDK). www.oracle.com/technetwork/java/javase/overview/index.html
- [27] Shrink that Footprint. www.shrinkthatfootprint.com/how-much-heating-energy-do-you-use

10. APPENDICES

10.1 JAVA IMPLEMENTATION OF THE EXPERT SYSTEM

10.1.1 INITIALIZE VARIABLES

```
private void jInitialActionPerformed(java.awt.event.ActionEvent evt) {  
    HeatNegotiation Test1 = new HeatNegotiation();  
    Test1.show();  
  
    jkW1.setText("0");  
    jkW2.setText("0");  
    jkW3.setText("0");  
    jkW4.setText("0");  
    jkW5.setText("0");  
    jTotal1.setText("0");  
  
    A1L=A1C=A1PC=A1SC=A1CAR=A1F=A1TV=A1W="0";  
    A2L=A2C=A2PC=A2SC=A2CAR=A2F=A2TV=A2W="0";  
    A3L=A3C=A3PC=A3SC=A3CAR=A3F=A3TV=A3W="0";  
    A4L=A4C=A4PC=A4SC=A4CAR=A4F=A4TV=A4W="0";  
    A5L=A5C=A5PC=A5SC=A5CAR=A5F=A5TV=A5W="0";  
    RULE1=RULE2=RULE3=RULE4=RULE5=RULE6=RULE7="0";  
    RH1=RH2=RH3=RH4=RH5= 0.0;  
  
    jLF1.setText("0");  
    jLF2.setText("0");  
    jLF3.setText("0");  
    jLF4.setText("0");  
    jLF5.setText("0");  
  
    jLog.setText("Day 1");  
    day=1;  
  
    new Timer(1000,taskPerformer).start();  
    hour = 0;
```

```

jInitial.setVisible(false);
jStep1.setVisible(false);
jStep2.setVisible(false);
jStep3.setVisible(false);
jStep4.setVisible(false);
jStep5.setVisible(false);
}

```

10.1.2 BUTTON FUNCTIONS

```

public String ChangeButton(java.awt.event.ActionEvent evt,javax.swing.JButton jButton,String
Text,String Deviceb,String Device) {
    String jText="0";
    String Status="0";
    if (Text.equals("1"))    {
        jText = "1";
        jButton.setBackground(Color.green);
        Status = "ON";
    }
    else if (Text.equals("0")) {
        jText = "0";
        jButton.setBackground(Color.red);
        Status = "OFF";
    }
    if (Deviceb.equals(Text)) {
    } else {
        jLog.setText(jLog.getText() + "\n" + Device + " is " + Status);
    }
    return jText;
}

```

```

public String InverseButton(java.awt.event.ActionEvent evt, javax.swing.JButton jButton, String
jText, String Device) {
    Color c = jButton.getBackground(); // Actual background color
    Color b = new Color(0,255,0); //Color green
    String Text;
    String Status;
    if (c.getRGB() == b.getRGB())
    {
        Text = "0";
        jButton.setBackground(Color.red);
        Status = "OFF";
    }
    else {
        Text = "1";
        jButton.setBackground(Color.green);
        Status = "ON";
    }
    jLog.setText(jLog.getText() + "\n" + Device + " is " + Status);
    return Text;
}

```

10.1.3 STEP FUNCTION

```

private void jStep1ActionPerformed(java.awt.event.ActionEvent evt) {
    String C = A1C;
    String L = A1L;
    String PC = A1PC;
    String SC = A1SC;
    String CAR = A1CAR;
    String T = jTime.getText();
    String F = A1F;
    String WF = jWeightF1.getText();
    String TV = A1TV;
    String W = A1W;
    String LF = jLF1.getText();
}

```

```

String RC = RandomNumber();
String RL = RandomNumber();
String RPC = RandomNumber();
String RSC = RandomNumber();
String RCAR = RandomNumber();
String RF = RandomNumber();
String RTV = RandomNumber();
String RW = RandomNumber(); //Creates random number that implies the accuracy of the
sensor in this moment

Rules(jRule1,jRule2,jRule3,jRule4,jRule5,jRule6,jRule7); //Gets the rules that will be
available to be fired

CalculateEnergy(evt,C,L,PC,SC,CAR,F,TV,W,jkW1);
CalculateTotal(evt,jTotal1,jkW1,jkW2,jkW3,jkW4,jkW5); // It calculates the energy of this
house and the total consumption

Query q2 = new
Query("rule("+RULE1+", "+RULE2+", "+RULE3+", "+RULE4+", "+RULE5+", "+RULE6+",
"+RULE7+", "+C+", "+RC+", "+CAR+", "+RCAR+", "+PC+", "+RPC+", "+SC+", "+RSC+", "+L+",
"+RL+", "+F+", "+RF+", "+WF+", "+TV+", "+RTV+", "+W+", "+RW+", "+LF+", "+T+", C1,CAR1,
L1,SC1,W1,LF1,S)");

java.util.Hashtable s1 = q2.oneSolution(); //Sends the variables to Prolog and it calculates the
new values according to the ruleset

jLF1.setText(s1.get("LF1").toString()); // Sets the level of the fridge

A1C = ChangeButton(evt,jC1,s1.get("C1").toString(),C,"Charger A1"); //Changes the status
of the device according to the new value given by Prolog
A1L = ChangeButton(evt,jL1,s1.get("L1").toString(),L,"Light A1");
A1SC = ChangeButton(evt,jSc1,s1.get("SC1").toString(),SC,"Screen A1");
A1CAR = ChangeButton(evt,jCar1,s1.get("CAR1").toString(),CAR,"CAR A1");
A1W = ChangeButton(evt,jW1,s1.get("W1").toString(),W,"Washing Machine A1");
}

```

10.1.4 CALCULATE ENERGY FUNCTIONS

```
public void CalculateTotal(java.awt.event.ActionEvent evt, javax.swing.JTextField
jTotal, javax.swing.JTextArea jkW1, javax.swing.JTextArea jkW2, javax.swing.JTextArea
j kW3, javax.swing.JTextArea j kW4, javax.swing.JTextArea j kW5) {

jTotal.setText(Integer.toString(Integer.parseInt(jkW2.getText())+Integer.parseInt(jkW1.getText())
+Integer.parseInt(j kW3.getText())+Integer.parseInt(j kW4.getText())+Integer.parseInt(j kW5.getTex
t())));

}

public void CalculateEnergy(java.awt.event.ActionEvent evt, String C, String L, String PC, String
SC, String CAR, String F, String TV, String W, javax.swing.JTextArea jText) {

    int T = 0;
    if (C.equals("0")) { }
    else { T = T + Cost.C ; }
    if (L.equals("0")) { }
    else { T = T + Cost.L ; }
    if (PC.equals("0")) { }
    else { T = T + Cost.Pc ; }
    if (SC.equals("0")) { }
    else { T = T + Cost.Sc ; }
    if (CAR.equals("0")) { }
    else { T = T + Cost.Car ; }
    if (F.equals("0")) { }
    else { T = T + Cost.F ; }
    if (TV.equals("0")) { }
    else { T = T + Cost.Tv ; }
    if (W.equals("0")) { }
    else { T = T + Cost.W ; }
    jText.setText(Integer.toString(T));

}
```

```

public String RandomNumber() {
    double a = 0.6 + 0.4*Math.random();
    a = (double)Math.round(a* 10000) / 10000 ;
    return (Double.toString(a));
}

```

10.1.5 STATE OF RULES FUNCTION

```

public void Rules(javax.swing.JToggleButton jRule1,javax.swing.JToggleButton
jRule2,javax.swing.JToggleButton jRule3,javax.swing.JToggleButton
jRule4,javax.swing.JToggleButton jRule5,javax.swing.JToggleButton
jRule6,javax.swing.JToggleButton jRule7) {
    RULE1 = RuleisOn(jRule1);
    RULE2 = RuleisOn(jRule2);
    RULE3 = RuleisOn(jRule3);
    RULE4 = RuleisOn(jRule4);
    RULE5 = RuleisOn(jRule5);
    RULE6 = RuleisOn(jRule6);
    RULE7 = RuleisOn(jRule7);
}

public String RuleisOn(javax.swing.JToggleButton jRule) {
    String Rule;
    if (jRule.isSelected()) {
        Rule = "1";
    } else {
        Rule = "0";
    }
    return Rule;
}

```


10.2 PROLOG IMPLEMENTATION OF THE RULE-SET

Here the entire code of the Prolog file will be displayed. It is divided into sections to make it more understandable.

10.2.1 GENERAL RULE

```
rule(RULE1,RULE2,RULE3,RULE4,RULE5,RULE6,RULE7,C,RC,CAR,RCAR,PC,RPC,SC,RS
C,L,RL,F,RF,WF,TV,RTV,W,RW,LF,T,C1,CAR1,L1,SC1,W1,LF1,S):-
    rule1(RULE1,CAR,RCAR,PC,RPC,SC,RSC,L,RL,TV,RTV,W,RW,S1),
    rule2(RULE2,L,RL,T,S2),
    rule3(RULE3,PC,RPC,SC,RSC,T,S3),
    rule4(RULE4,F,RF,WF,LF,LF1,S4),
    rule5(RULE5,TV,RTV,L,RL,T,S5),
    rule6(RULE6,W,RW,T,S6),
    rule7(RULE7,C,RC,T,S7),
    decision(S1,S2,S3,S4,S5,S6,S7,C,CAR,L,SC,W,C1,CAR1,L1,SC1,W1,LF1,S).
```

10.2.2 RULE-SET

```
rule1(RULE1,CAR,RCAR,PC,RPC,SC,RSC,L,RL,TV,RTV,W,RW,S1):- %Surpassing kW
    S1 is (0.9*CAR*RCAR+0.7*W*RW+0.6*TV*RTV+0.2*PC*RPC+0.2*SC*RSC+
    0.3*L*RL)*(1/2.9)*RULE1.
rule2(RULE2,L,RL,T,S2):- %Light on during sunlight hours
    calctime(T,TL),
    S2 is (0.6*RL+0.9*TL)*L*(1/1.5)*RULE2.
rule3(RULE3,PC,RPC,SC,RSC,T,S3):- %Screen On and computer Off
    calctimeS(T,TS),
    S3 is (0.9*RPC+0.9*RSC+0.3*TS)*(1/2.1)*(1-PC)*SC*RULE3.
rule4(RULE4,F,RF,WF,LF,LF1,S4):- %calculates level of the fridge
    calcweight(WF,LF1,F),
    S4 is (0.9*F*RF+(LF1-LF)*0.3)*(1/1.2)*F*RULE4.
rule5(RULE5,TV,RTV,L,RL,T,S5):- %control tv and light
    calctvtime(T,TTV),
    S5 is (0.9*RTV+0.8*RL+0.7*TTV)*TV*L*(1/2.4)*RULE5.
```

rule6(RULE6,W,RW,T,S6) :- %Control Washing Machine

calcwtime(T,TW),

S6 is $(0.9*TW+0.9*RW*W)*RULE6*TW*(1/1.8)$.

rule7(RULE7,C,RC,T,S7) :- %Control Phone Charger

calcctime(T,TC),

S7 is $(0.9*TC+0.9*RC)*(1-C)*RULE7*(1/1.8)$.

10.2.3 DECISION FUNCTION

decision(S1,S2,S3,S4,S5,S6,S7,C,CAR,L,SC,W,C1,CAR1,L1,SC1,W1,LF,S):-

(S1 < 0.6 , S2 < 0.3 , S3 < 0.3 , S4 < 0.3 , S5 < 0.3 , S6 < 0.3 , S7 < 0.3 ->

CAR1=CAR, L1=L, SC1=SC , W1=W , C1=C);

(S1 := max(S1,S2) , S1 := max(S1,S3) , S1 := max(S1,S4) , S1 := max(S1,S5), S1
:= max(S1,S6), S1 := max(S1,S7) ->

S is S1 , CAR1=0 , L1=L , SC1=SC , W1=W , C1=C);

(S2 := max(S1,S2) , S2 := max(S3,S2) , S2 := max(S2,S4) , S2 := max(S2,S5), S2 =
:= max(S2,S6), S2 := max(S2,S7) ->

S is S2 , L1=0 , CAR1=CAR , SC1=SC , W1=W , C1=C);

(S3 := max(S3,S2) , S3 := max(S1,S3) , S3 := max(S3,S4) , S3 := max(S3,S5), S3
:= max(S3,S6), S3 := max(S3,S7) ->

S is S3 , CAR1=CAR , L1=L, SC1=0 , W1=W , C1=C);

(S4 := max(S4,S1) , S4 := max(S4,S2) , S4 := max(S4,S3) , S4 := max(S4,S5), S4
:= max(S4,S6), S4 := max(S4,S7) ->

S is S4 , LF=LF, CAR1=CAR, SC1=SC, L1=L , W1=W , C1=C);

(S5 := max(S5,S1) , S5 := max(S5,S2) , S5 := max(S5,S3) , S5 := max(S5,S4), S5
:= max(S5,S6), S5 := max(S5,S7) ->

S is S5 , L1=0 , CAR1=CAR , SC1=SC , W1=W , C1=C);

(S6 := max(S6,S1) , S6 := max(S6,S2) , S6 := max(S6,S3) , S6 := max(S6,S4), S6
:= max(S6,S5), S6 := max(S6,S7) ->

S is S6 , W1=0 , L1=L , CAR1=CAR , SC1=SC , C1=C);

(S7 := max(S7,S1) , S7 := max(S7,S2) , S7 := max(S7,S3) , S7 := max(S7,S4), S7
:= max(S7,S5), S7 := max(S6,S7) ->

S is S7 , C1=1 , W1=W , L1=L , CAR1=CAR , SC1=SC).

10.2.4 OTHER FUNCTIONS

calctimeS(T,TS):- %This function find the night hours when we are sleeping

```
( T >=3 , T <= 7 -> TS is 1 );  
( T >=0 , T <= 2 -> TS is T/3 );  
( T >=8 , T <= 10 -> TS is (10-T)/3 );  
( TS is 0 ).
```

calctime(T,TL):- %This function finds the Sun hours when we don't need light

```
( T >=9 , T <= 16 -> TL is 1 );  
( T >=6 , T <=8 -> TL is (T-6)/3 );  
( T >=17 , T <= 19 -> TL is (19-T)/3 );  
( TL is 0 ).
```

calcweight(WF,LF1,F):- %This function finds the appropriate level of the fridge

```
( WF <= 5 , F=1 -> LF1=1 );  
( WF > 5 , WF <= 8 , F=1 -> LF1=2 );  
( WF > 8 , WF <= 13 , F=1 -> LF1=3 );  
( WF > 13 , WF <= 20 , F=1 -> LF1=4 );  
( WF > 20 , F=1 -> LF1=5 );  
( F=0 -> LF1=1 ).
```

calctvtime(T,TTV):- %This function gives 1 if the time is close to late night (movie time)

```
( T >=22, T <= 1 -> TTV is 1 );  
( T >=2 , T <= 3 -> TTV is (3-T)/2 );  
( T >=20 , T <= 21 -> TTV is (T-20)/2 );  
( TTV is 0 ).
```

calcwtime(T,TW):-

```
( T >=1 , T <=8 -> TW is 1 );  
( TW is 0 ).
```

calcctime(T,TC):-

```
( T >=0 , T <=6 -> TC is 1 );  
( TC is 0 ).
```

10.3 JAVA IMPLEMENTATION OF THE HEAT CONTROL SYSTEM

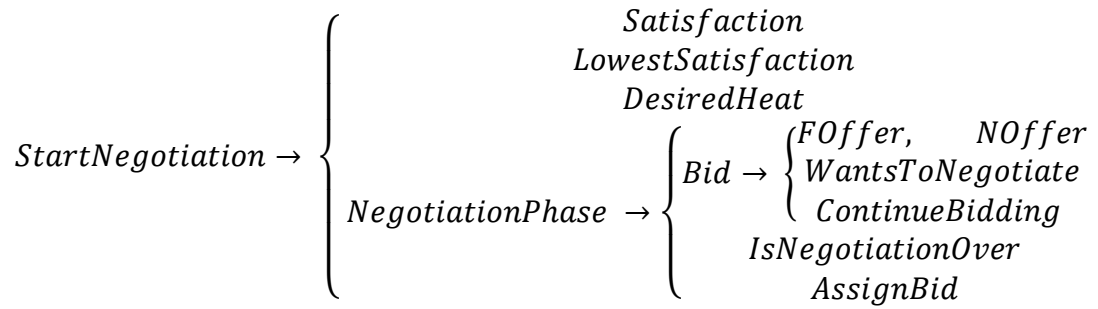


Figure 10.3 Heat Control System Diagram

In the Figure 10.3 there is a diagram displaying what functions are inside the negotiation function. The StartNegotiation function includes four functions inside it. The last of these functions includes three more different functions inside it.

10.3.1 START NEGOTIATION FUNCTION

```
public static int StartNegotiation(javax.swing.JTextArea jA1,javax.swing.JTextArea
jA2,javax.swing.JTextArea jA3,javax.swing.JTextArea jA4,javax.swing.JTextArea jA5) {
    int LS = LowestSatisfaction(Satisfaction(HeatControl.A1,1),Satisfaction(HeatControl.A2,2)
,Satisfaction(HeatControl.A3,3),Satisfaction(HeatControl.A4,4),Satisfaction(HeatControl.A5,5));
    jA1.setText(Integer.toString(A1));
    jA2.setText(Integer.toString(A2));
    jA3.setText(Integer.toString(A3));
    jA4.setText(Integer.toString(A4));
    jA5.setText(Integer.toString(A5));
    if (LS==1) { //Starts the negotiation with agent with the lowest satisfaction (LS)
        NegotiationPhase(A2,2,A3,3,A4,4,A5,5,DesiredHeat(1),1,jA2,jA3,jA4,jA5,jA1,A1);
    } else if (LS==2) {
        NegotiationPhase(A1,1,A3,3,A4,4,A5,5,DesiredHeat(2),2,jA1,jA3,jA4,jA5,jA2,A2);
    } else if (LS==3) {
        NegotiationPhase(A1,1,A2,2,A4,4,A5,5,DesiredHeat(3),3,jA1,jA2,jA4,jA5,jA3,A3);
    } else if (LS==4) {
        NegotiationPhase(A1,1,A2,2,A3,3,A5,5,DesiredHeat(4),4,jA1,jA2,jA3,jA5,jA4,A4);
    } else {
        NegotiationPhase(A1,1,A2,2,A3,3,A4,4,DesiredHeat(5),5,jA1,jA2,jA3,jA4,jA5,A5);
    }
}
```

```

    }
    return LS;
}

```

10.3.2 SATISFACTION FUNCTIONS AND DESIRED HEAT

```

public static int Satisfaction(int H,int A) { //Calculates the agent's satisfaction
    int Sat = (H-(2000-A*100))/20;
    return Sat;
}

```

```

public static int LowestSatisfaction(int A1,int A2,int A3,int A4,int A5) {
    int LA;
    if ((A1<=A2) && (A1<=A3) && (A1<=A4) && (A1<=A5)) { LA=1;
    } else if ((A2<=A1) && (A2<=A3) && (A2<=A4) && (A2<=A5)) { LA=2;
    } else if ((A3<=A1) && (A3<=A2) && (A3<=A4) && (A3<=A5)) { LA=3;
    } else if ((A4<=A1) && (A4<=A2) && (A4<=A3) && (A4<=A5)) { LA=4;
    } else { LA=5; }
    return LA;
}

```

```

public static int DesiredHeat(int A) {
    int DH = (75*20+(2000-100*A));
    return DH;
}

```

10.3.3 NEGOTIATION PHASE FUNCTIONS

```

public static void NegotiationPhase(int A1,int nA1,int A2,int nA2,int A3,int nA3,int A4,int
nA4,int DesiredH,int DA,javax.swing.JTextArea jA1,javax.swing.JTextArea
jA2,javax.swing.JTextArea jA3,javax.swing.JTextArea jA4,javax.swing.JTextArea jA5,int A5){
    int DesiredHeat=DesiredH;
    int N;

    jA5.setText("Agent " + DA + " requests negotiation");
}

```

```

jA1.setText("Initial state : " + A1);
jA2.setText("Initial state : " + A2);
jA3.setText("Initial state : " + A3);
jA4.setText("Initial state : " + A4);
jA5.setText(jA5.getText() + "\n" + "Initial state : " + A5);

```

```

double Bid1 = Bid(A1,1,FOffer(Satisfaction(A1,nA1)),nA1);
double Bid2 = Bid(A2,1,FOffer(Satisfaction(A2,nA2)),nA2);
double Bid3 = Bid(A3,1,FOffer(Satisfaction(A3,nA3)),nA3);
double Bid4 = Bid(A4,1,FOffer(Satisfaction(A4,nA4)),nA4);

```

```

jA1.setText(jA1.getText() + "\n" + "Bid 1 : " + Integer.toString((int)Bid1));
jA2.setText(jA2.getText() + "\n" + "Bid 1 : " + Integer.toString((int)Bid2));
jA3.setText(jA3.getText() + "\n" + "Bid 1 : " + Integer.toString((int)Bid3));
jA4.setText(jA4.getText() + "\n" + "Bid 1 : " + Integer.toString((int)Bid4));
jA5.setText(jA5.getText() + "\n" + "Bid 1 : " + Integer.toString(DesiredHeat));

```

```

if (IsNegotiationOver(Bid1,Bid2,Bid3,Bid4,DesiredHeat)) {

```

```

    N=1;

```

```

} else {

```

```

    N=2;

```

```

    Bid1 = Bid((int)Bid1,N,NOffer(Satisfaction((int)Bid1,nA1),N),nA1);

```

```

    Bid2 = Bid((int)Bid2,N,NOffer(Satisfaction((int)Bid2,nA2),N),nA2);

```

```

    Bid3 = Bid((int)Bid3,N,NOffer(Satisfaction((int)Bid3,nA3),N),nA3);

```

```

    Bid4 = Bid((int)Bid4,N,NOffer(Satisfaction((int)Bid4,nA4),N),nA4);

```

```

    DesiredHeat=DesiredHeat-100;

```

```

jA1.setText(jA1.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid1));

```

```

jA2.setText(jA2.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid2));

```

```

jA3.setText(jA3.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid3));

```

```

jA4.setText(jA4.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid4));

```

```

jA5.setText(jA5.getText() + "\n" + "Bid " + N + " : " + Integer.toString(DesiredHeat));

```

```

while(!IsNegotiationOver(Bid1,Bid2,Bid3,Bid4,DesiredHeat) && N<7) {

```

```

        Bid1 = Bid((int)Bid1,N,NOffer(Satisfaction((int)Bid1,nA1),N),nA1);
        Bid2 = Bid((int)Bid2,N,NOffer(Satisfaction((int)Bid2,nA2),N),nA2);
        Bid3 = Bid((int)Bid3,N,NOffer(Satisfaction((int)Bid3,nA3),N),nA3);
        Bid4 = Bid((int)Bid4,N,NOffer(Satisfaction((int)Bid4,nA4),N),nA4);

        DesiredHeat=DesiredHeat-50;
        N=N+1;

        jA1.setText(jA1.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid1));
        jA2.setText(jA2.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid2));
        jA3.setText(jA3.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid3));
        jA4.setText(jA4.getText() + "\n" + "Bid " + N + " : " + Integer.toString((int)Bid4));
        jA5.setText(jA5.getText() + "\n" + "Bid " + N + " : " + Integer.toString(DesiredHeat));
    }

}

if (N>7) {
    } else {
        AssignBid((int)Bid1,(int)Bid2,(int)Bid3,(int)Bid4,DesiredHeat,DA);
    }
}

public static boolean IsNegotiationOver(double Bid1,double Bid2,double Bid3,double
Bid4,int DesiredH) {
    boolean Over=false;
    if ((int)Bid1+(int)Bid2+(int)Bid3+(int)Bid4+DesiredH<=HeatControl.Limit) {
        Over=true;
    } else {
    }
    return Over;
}

```

10.3.4 BID FUNCTIONS

```
public static double Bid(int Heat,int NOffer,double Percentage,int A) {
    double Bid;
    if (NOffer==1) {
        Bid=Heat*((1-Percentage)-(Percentage*0.4-0.2)*(Math.random()/100));
    } else {
        Bid=Heat*(1-Percentage-(Math.random()/100));
    }
    if (WantsToNegotiate()==1) { //If the agent does not want to negotiate -> bid = constant
        Bid = Heat;
    } else {
    }
    Bid = ContinueBidding(Satisfaction((int)Bid,A),Bid,Heat);
    return Bid;
}

public static double FOffer(int Sat) { //Bid percentage for the first offer
    double Percent=4.0;
    if ((Sat<55) && (Sat>=50)) {
        Percent=1.0;
    } else if ((Sat<60) && (Sat>=55)) {
        Percent=1.5;
    } else if ((Sat<65) && (Sat>=60)) {
        Percent=2.0;
    } else if ((Sat<70) && (Sat>=65)) {
        Percent=2.5;
    } else if ((Sat<75) && (Sat>=70)) {
        Percent=3.0;
    } else if ((Sat<80) && (Sat>=75)) {
        Percent=3.5;
    } else {
    }
    return (Percent/100);
}
```



```
public static double NOffer(int Sat,int N) { //Bid percentage for the second,third and so on  
offers
```

```
    double NOffer= 1;  
    if ((Sat<55) && (Sat>=50)) {  
        NOffer=0.2;  
    } else if ((Sat<60) && (Sat>=55)) {  
        NOffer=0.4;  
    } else if ((Sat<65) && (Sat>=60)) {  
        NOffer=0.6;  
    } else if ((Sat<70) && (Sat>=65)) {  
        NOffer=0.8;  
    } else if ((Sat<75) && (Sat>=70)) {  
        NOffer=1.0;  
    } else if ((Sat<80) && (Sat>=75)) {  
        NOffer=1.2;  
    } else {  
    }  
    return (NOffer/100);  
}
```

```
public static int WantsToNegotiate(){ //Gives a 1 95% of the times, if not, zero.  
    int Neg;  
    if (Math.random()>0.05) { Neg=0; }  
    else { Neg=1; }  
    return Neg;  
}
```

```
public static double ContinueBidding(int Satisfaction,double Bid,int Heat) {  
    if (Satisfaction<45) { //If the agent's satisfcation goes below 45, he will stop bidding  
        return Heat;  
    } else  
        return Bid;  
}
```

```

public static void AssignBid(int Bid1,int Bid2,int Bid3,int Bid4,int DesiredH,int DA){
    if (DA==1) {
        HeatControl.A1=DesiredH;
        HeatControl.A2=Bid1;
        HeatControl.A3=Bid2;
        HeatControl.A4=Bid3;
        HeatControl.A5=Bid4;
    } else if (DA==2) {
        HeatControl.A1=Bid1;
        HeatControl.A2=DesiredH;
        HeatControl.A3=Bid2;
        HeatControl.A4=Bid3;
        HeatControl.A5=Bid4;
    } else if (DA==3) {
        HeatControl.A1=Bid1;
        HeatControl.A2=Bid2;
        HeatControl.A3=DesiredH;
        HeatControl.A4=Bid3;
        HeatControl.A5=Bid4;
    } else if (DA==4) {
        HeatControl.A1=Bid1;
        HeatControl.A2=Bid2;
        HeatControl.A3=Bid3;
        HeatControl.A4=DesiredH;
        HeatControl.A5=Bid4;
    } else {
        HeatControl.A1=Bid1;
        HeatControl.A2=Bid2;
        HeatControl.A3=Bid3;
        HeatControl.A4=Bid4;
        HeatControl.A5=DesiredH;
    }
}
}

```

10.4 JAVA IMPLEMENTATION OF THE ONE-BID SYSTEM

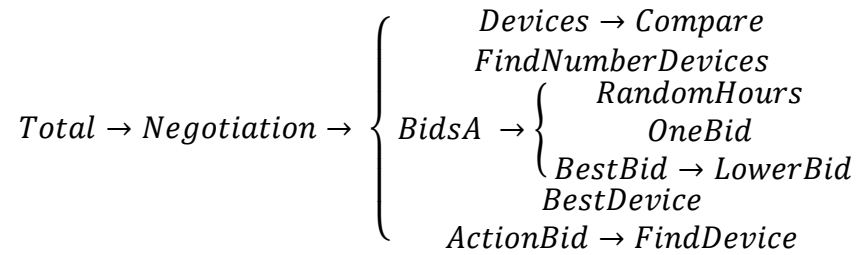


Diagram displaying what functions are inside the main function. Total includes the function Negotiation, which includes all other five functions. And each one of these five includes other functions.

10.4.1 MAIN FUNCTION

```
public String Total(int Ad,java.awt.event.ActionEvent evt,javax.swing.JButton jButton,String
jText,javax.swing.JTextField jTotal,int i,String Device) {
    String Total="0";
    int a = Integer.parseInt(jTotal.getText()) + i;
    if ((a>Cost.Total) && (jText.equals("0")) && (jToggleButton1.isSelected())) { //Starts the
negotiation if the user can't turn on the device
        int A1 = Integer.parseInt(jkW1.getText());
        int A2 = Integer.parseInt(jkW2.getText());
        int A3 = Integer.parseInt(jkW3.getText());
        int A4 = Integer.parseInt(jkW4.getText());
        int A5 = Integer.parseInt(jkW5.getText());

        Total = Negotiation(evt,Ad,i,A1,A2,A3,A4,A5,jButton,jText,Device);
        if (Total.equals("0")) {
            } else {
                jToggleButton1.setSelected(false);
            }
        } else if ((a>Cost.Total) && (jText.equals("0"))) { //You need to click on the jToggleButton1
to start the negotiation
            Total = jText;
        } else {
```

```

        Total = InverseButton(evt,jButton,jText,Device);
    }
    return Total;
}

```

10.4.2 NEGOTIATION FUNCTION

```

public String Negotiation(java.awt.event.ActionEvent evt,int Ad,int i,int A1,int A2,int A3,int
A4,int A5,javax.swing.JButton jButton,String jText,String Device) {
    String D1,D2,D3,D4,D5;
    int N1,N2,N3,N4,N5,NB;
    String ND;
    if (Ad==1) { //Starts the negotiation according to whom is the demanding agent (Ad)
        D2 = Devices(evt,i,A2,A2C,A2L,A2PC,A2SC,A2CAR,A2F,A2TV,A2W);
        N2=FindNumberDevices(evt,D2);
        D3 = Devices(evt,i,A3,A3C,A3L,A3PC,A3SC,A3CAR,A3F,A3TV,A3W);
        N3=FindNumberDevices(evt,D3);
        D4 = Devices(evt,i,A4,A4C,A4L,A4PC,A4SC,A4CAR,A4F,A4TV,A4W);
        N4=FindNumberDevices(evt,D4);
        D5 = Devices(evt,i,A5,A5C,A5L,A5PC,A5SC,A5CAR,A5F,A5TV,A5W);
        N5=FindNumberDevices(evt,D5);
        NB=BidsA1(evt,N2,N3,N4,N5,D2,D3,D4,D5);
        ND=BestDevice(NB,2,D2,3,D3,4,D4,5,D5);
    } else if (Ad==2) {
        D1 = Devices(evt,i,A1,A1C,A1L,A1PC,A1SC,A1CAR,A1F,A2TV,A2W);
        N1=FindNumberDevices(evt,D1);
        D3 = Devices(evt,i,A3,A3C,A3L,A3PC,A3SC,A3CAR,A3F,A3TV,A3W);
        N3=FindNumberDevices(evt,D3);
        D4 = Devices(evt,i,A4,A4C,A4L,A4PC,A4SC,A4CAR,A4F,A4TV,A4W);
        N4=FindNumberDevices(evt,D4);
        D5 = Devices(evt,i,A5,A5C,A5L,A5PC,A5SC,A5CAR,A5F,A5TV,A5W);
        N5=FindNumberDevices(evt,D5);
        NB=BidsA2(evt,N1,N3,N4,N5,D1,D3,D4,D5);
        ND=BestDevice(NB,1,D1,3,D3,4,D4,5,D5);
    } else if (Ad==3) {

```

```

    D2 = Devices(evt,i,A2,A2C,A2L,A2PC,A2SC,A2CAR,A2F,A2TV,A2W);
    N2=FindNumberDevices(evt,D2);
    D1 = Devices(evt,i,A1,A1C,A1L,A1PC,A1SC,A1CAR,A1F,A1TV,A1W);
    N1=FindNumberDevices(evt,D1);
    D4 = Devices(evt,i,A4,A4C,A4L,A4PC,A4SC,A4CAR,A4F,A4TV,A4W);
    N4=FindNumberDevices(evt,D4);
    D5 = Devices(evt,i,A5,A5C,A5L,A5PC,A5SC,A5CAR,A5F,A5TV,A5W);
    N5=FindNumberDevices(evt,D5);
    NB=BidsA3(evt,N1,N2,N4,N5,D1,D2,D4,D5);
    ND=BestDevice(NB,1,D1,2,D2,4,D4,5,D5);
} else if (Ad==4) {
    D2 = Devices(evt,i,A2,A2C,A2L,A2PC,A2SC,A2CAR,A2F,A2TV,A2W);
    N2=FindNumberDevices(evt,D2);
    D3 = Devices(evt,i,A3,A3C,A3L,A3PC,A3SC,A3CAR,A3F,A3TV,A3W);
    N3=FindNumberDevices(evt,D3);
    D1 = Devices(evt,i,A1,A1C,A1L,A1PC,A1SC,A1CAR,A1F,A1TV,A1W);
    N1=FindNumberDevices(evt,D1);
    D5 = Devices(evt,i,A5,A5C,A5L,A5PC,A5SC,A5CAR,A5F,A5TV,A5W);
    N5=FindNumberDevices(evt,D5);
    NB=BidsA4(evt,N1,N2,N3,N5,D1,D2,D3,D5);
    ND=BestDevice(NB,1,D1,3,D3,2,D2,5,D5);
} else {
    D2 = Devices(evt,i,A2,A2C,A2L,A2PC,A2SC,A2CAR,A2F,A2TV,A2W);
    N2=FindNumberDevices(evt,D2);
    D3 = Devices(evt,i,A3,A3C,A3L,A3PC,A3SC,A3CAR,A3F,A3TV,A3W);
    N3=FindNumberDevices(evt,D3);
    D4 = Devices(evt,i,A4,A4C,A4L,A4PC,A4SC,A4CAR,A4F,A4TV,A4W);
    N4=FindNumberDevices(evt,D4);
    D1 = Devices(evt,i,A1,A1C,A1L,A1PC,A1SC,A1CAR,A1F,A1TV,A1W);
    N1=FindNumberDevices(evt,D1);
    NB=BidsA5(evt,N1,N2,N3,N4,D1,D2,D3,D4);
    ND=BestDevice(NB,1,D1,3,D3,4,D4,2,D2);
}
return ActionBid(evt,NB,ND,jButton,jText,Device);
}

```

10.4.3 DEVICE FUNCTIONS

```
public String Devices (java.awt.event.ActionEvent evt,int i,int A1,String jCh,String jL,String
jPc,String jSc,String jCar,String jF,String jTv,String jW) {
    int C = Integer.parseInt(jCh);
    int L = Integer.parseInt(jL);
    int PC = Integer.parseInt(jPc);
    int SC = Integer.parseInt(jSc);
    int CAR = Integer.parseInt(jCar);
    int F = Integer.parseInt(jF);
    int TV = Integer.parseInt(jTv);
    int W = Integer.parseInt(jW);

    String D = "0";

    if (Compare(C,A1,Cost.C,i)) {      D = "C";
    } else if (Compare(SC,A1,Cost.Sc,i)) {  D = "SC"; //20
    } else if (Compare(PC,A1,Cost.Pc,i)) {  D = "PC"; //50
    } else if (Compare(L,A1,Cost.L,i)) {    D = "L"; //60
    } else if (Compare(L,A1,Cost.Sc + Cost.Pc,i)) {  D = "SC+PC"; //70
    } else if (Compare(TV,A1,Cost.Tv,i)) {   D = "TV"; //500
    } else if (Compare(F,A1,Cost.F,i)) {    D = "F"; //600
    } else if (Compare(W,A1,Cost.W,i)) {    D = "W"; //800
    } else if (Compare(CAR,A1,Cost.Car,i)) { D = "CAR"; //3000
    } else {
    }
    return D;
}

public boolean Compare(int D,int Agent,int Cost1,int i) {
    int total = Integer.parseInt(jTotal1.getText());
    return ((D==1) && (total-Cost1+i <= Cost.Total));
}

public int FindNumberDevices(java.awt.event.ActionEvent evt, String D) {
```

```

int N=1;
if (D.equals("SC+PC")) {
    N=2;
} else { }
return N;
}

```

```

public String BestDevice(int WB,int B1,String ND1,int B2,String ND2,int B3,String ND3,int
B4,String ND4) {
    String ND;
    if (WB==B1 ) { ND=ND1;
    } else if (WB==B2) { ND=ND2;
    } else if (WB==B3) { ND=ND3;
    } else { ND=ND4;
    }
    return ND;
}

```

10.4.4 BID FUNCTIONS

```

public int BidsA1(java.awt.event.ActionEvent evt,int N1,int N2,int N3,int N4,String D2,String
D3,String D4,String D5) { //Returns 1..4 (Best Bid)
    RH2=RandomHours();
    RH3=RandomHours();
    RH4=RandomHours();
    RH5=RandomHours();
    double B2 = OneBid(RH2,N1,1,ScoreDesire(0,pref,2),jBid2,D2,jA2,Demand.A2,1,2);
    double B3 = OneBid(RH3,N2,1,ScoreDesire(0,pref,3),jBid3,D3,jA3,Demand.A3,1,3);
    double B4 = OneBid(RH4,N3,2,ScoreDesire(0,pref,4),jBid4,D4,jA4,Demand.A4,1,4);
    double B5 = OneBid(RH5,N4,2,ScoreDesire(0,pref,5),jBid5,D5,jA5,Demand.A5,1,5);
    return BestBid(evt,B2,2,Demand.A2,B3,3,Demand.A3,B4,4,Demand.A4,B5,5,Demand.A5,
jReq1,jA1,1);
}

```

```

public double RandomHours() {
    return ((double)Math.round(1 + 3.0*Math.random();* 100) / 100);
}

public double OneBid(double H,int N,int Ne,int Action,javax.swing.JTextField jBid,String
D,javax.swing.JTextArea jA,String OBid,int SA,int BA){
    double Bid;
    if (jBid.getText().equals("")) {
        Bid = 0.2*H+0.05*(double)N+0.1*(double)Ne+0.65*(double)Action;
    } else {
        Bid = Double.parseDouble(jBid.getText());
    }
    double BidT = (double)Math.round(Bid* 1000) / 1000 ;
    if (D.equals("0")) {
        BidT=0.0;
    } else {
        jA.setText("Agent " + Integer.toString(BA) + " demands to Agent " +
Integer.toString(SA) + "\n" + "You will "+ OBid );
    }
    jBid.setText(Double.toString(BidT));
    return Bid;
}

public int BestBid(java.awt.event.ActionEvent evt,double B1,int NB1,String D1,double B2,int
NB2,String D2,double B3,int NB3,String D3,double B4,int NB4,String
D4,javax.swing.JToggleButton jReq,javax.swing.JTextArea jA,int A) {
    int NB=0;
    String ND="0";

    if (LowestBid(B1,B2,B3,B4)) { NB=NB1; ND=D1;
    } else if (LowestBid(B2,B1,B3,B4)) { NB=NB2; ND=D2;
    } else if (LowestBid(B3,B1,B2,B4)) { NB=NB3; ND=D3;
    } else if (LowestBid(B4,B1,B2,B3)) { NB=NB4; ND=D4;
    }

    if (!jReq.isSelected()) { //if jReq not selected it doesn't choose a winning bid
        NB=0;
    }
}

```



```

jLog.setText(jLog.getText() + "\n" + "Agent " + A + " starts negotiation");
jButton1ActionPerformed(evt);
} else {
    jReq.setSelected(false);
    jLog.setText(jLog.getText() + "\n" + "Agent " + A + " negotiates with agent " + NB);
    jA.setText("Agent " + A + " will " + ND);
    changecolor(NB,1); //Highlights the agent that is taking part in the negotiation
}
return NB; //returns winning bid and device
}

```

```

public boolean LowestBid(double B1,double B2,double B3,double B4) {
    boolean b1win=false;
    if (B1!=0 && B2!=0 && B3!=0 && B4!=0) {
        if (B1<B2 && B1<B3 && B1<B4) {
            b1win=true;
        }
    } else if (B1==0) {
    } else if (B2==0 && B3==0 && B4==0) { b1win=true;
    } else {
        if (B2==0) {
            if (B3==0) {
                if (B1<B4) { b1win=true; }
            } else if (B4==0) {
                if (B1<B3) { b1win=true; }
            } else if (B1<B3 && B1<B4) { b1win=true; }
        } else if (B3==0) {
            if (B4==0) {
                if (B1<B2) {b1win=true; }
            }
        } else if (B1<B2 && B1<B4) { b1win=true; }
    } else if (B4==0) {
        if (B1<B2 && B1<B3) {b1win=true; }
    }
}
}

```

```

    return b1win;
}

```

10.4.5 SOLUTION FUNCTIONS

```

public String ActionBid(java.awt.event.ActionEvent evt,int B,String D,javax.swing.JButton
jButton,String jText,String Device) {
    String Action;
    if (B==1) { FindDevice1(evt,D); Action=InverseButton(evt,jButton,jText,Device);
    } else if (B==2) { FindDevice2(evt,D); Action=InverseButton(evt,jButton,jText,Device);
    } else if (B==3) { FindDevice3(evt,D); Action=InverseButton(evt,jButton,jText,Device);
    } else if (B==4) { FindDevice4(evt,D); Action=InverseButton(evt,jButton,jText,Device);
    } else if (B==5) { FindDevice5(evt,D); Action=InverseButton(evt,jButton,jText,Device);
    } else { Action="0";
    }
    return Action;
}

```

```

public void FindDevice1(java.awt.event.ActionEvent evt, String D) {
    switch (D) {
        case "C": A1C = InverseButton(evt,jC1,A1C,"Charger A1"); break;
        case "L": A1L = InverseButton(evt,jL1,A1L,"Light A1"); break;
        case "PC": A1PC = InverseButton(evt,jPc1,A1PC,"PC A1"); break;
        case "SC": A1SC = InverseButton(evt,jSc1,A1SC,"Screen A1"); break;
        case "CAR": A1CAR = InverseButton(evt,jCar1,A1CAR,"Car A1"); break;
        case "F": A1F = InverseButton(evt,jF1,A1F,"Fridge A1"); break;
        case "SC+PC": A1SC = InverseButton(evt,jSc1,A1SC,"Screen A1");
            A1PC = InverseButton(evt,jPc1,A1PC,"PC A1"); break;
        case "TV": A1TV = InverseButton(evt,jTv1,A1TV,"Television A1"); break;
        case "W": A1W = InverseButton(evt,jW1,A1W,"Washing Machine A1"); break;
    }
}

```

10.5 SCREEN CAPTURES OF THE SIMULATION

10.5.1 SCREEN CAPTURE OF THE MAIN SIMULATOR

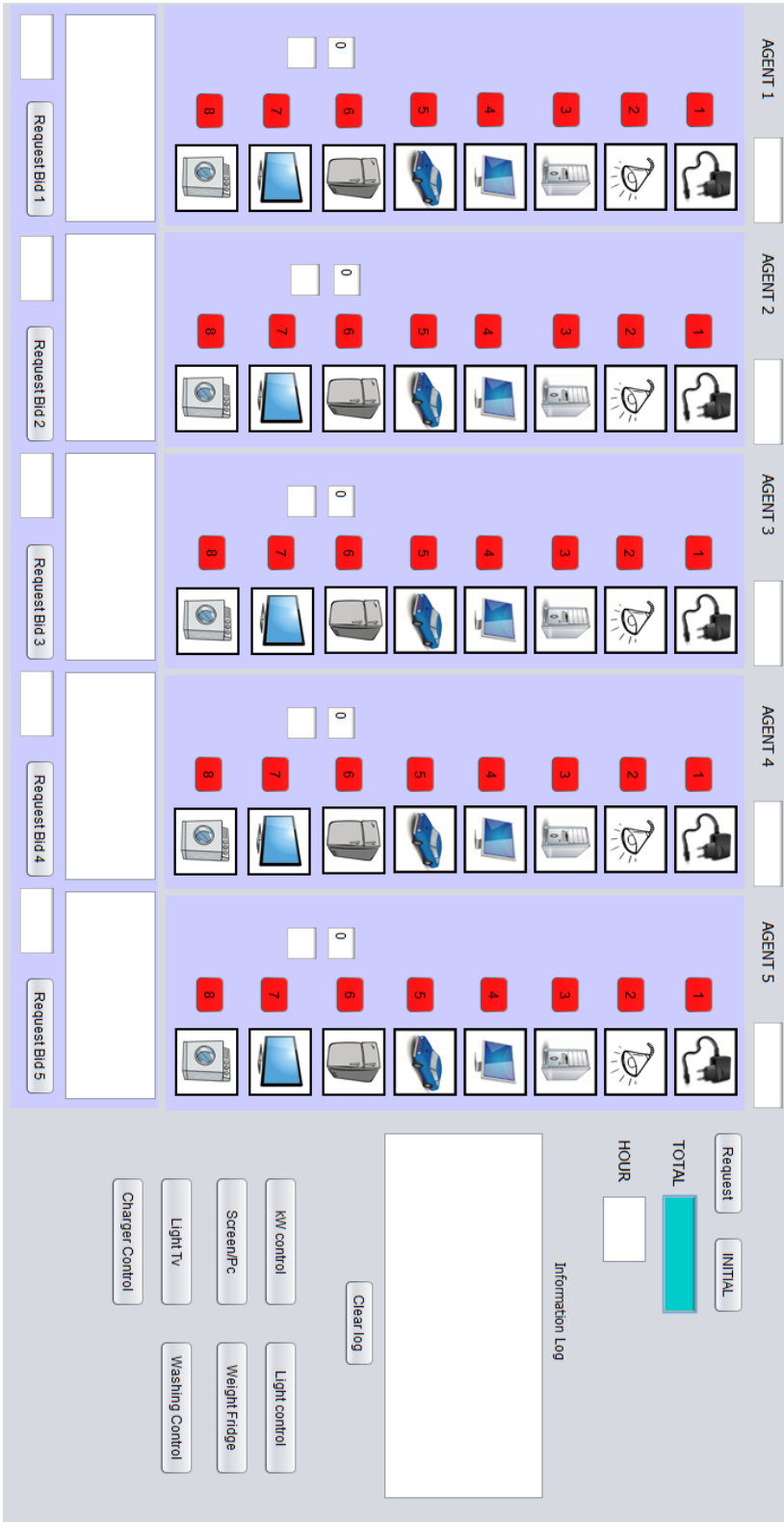


Figure 10.5.2. Initial state of the main window of the simulation

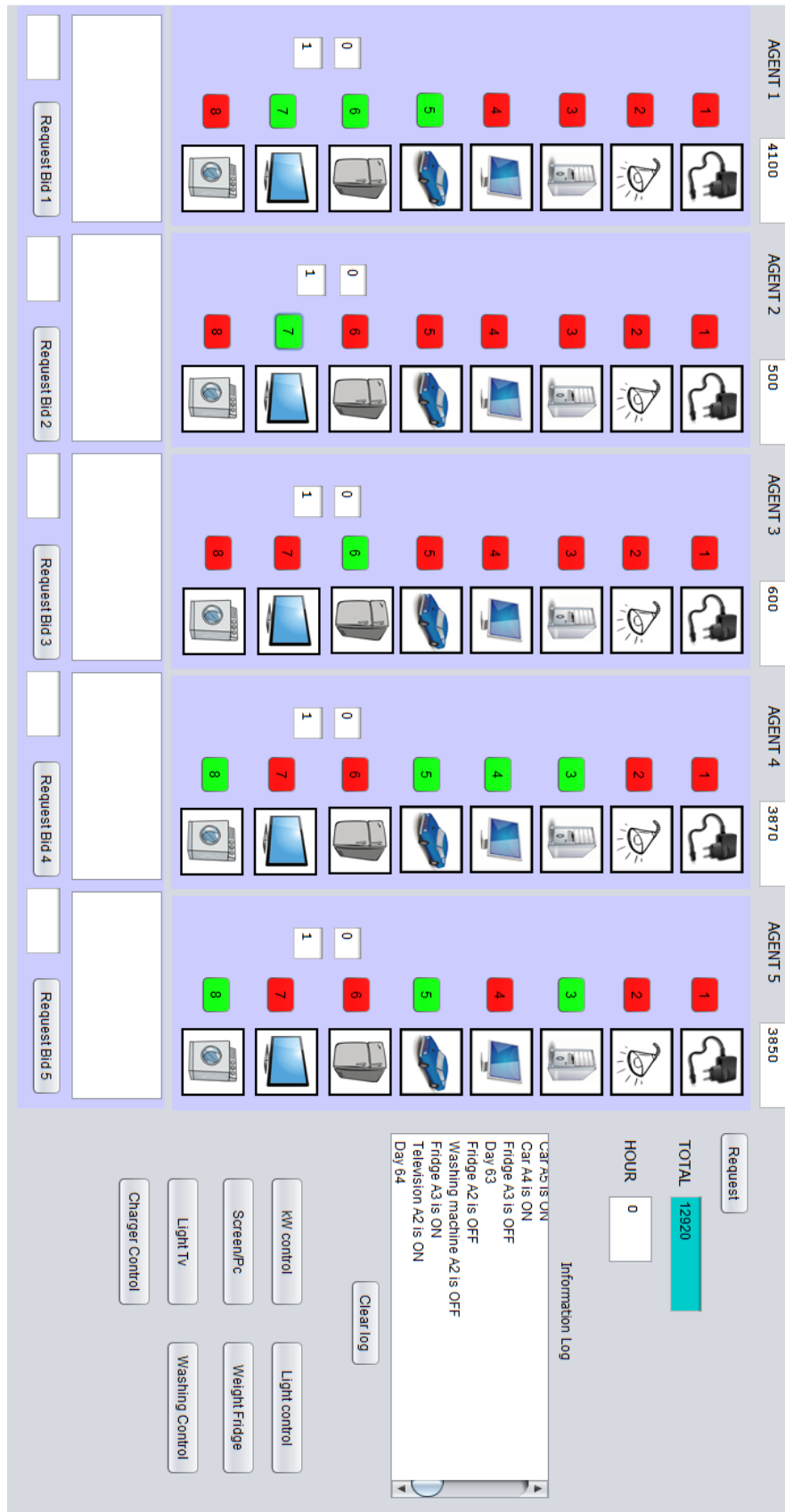


Figure 10.5.2 Main window of the simulation

10.5.2 SCREEN CAPTURE OF THE HEAT CONTROL WINDOW

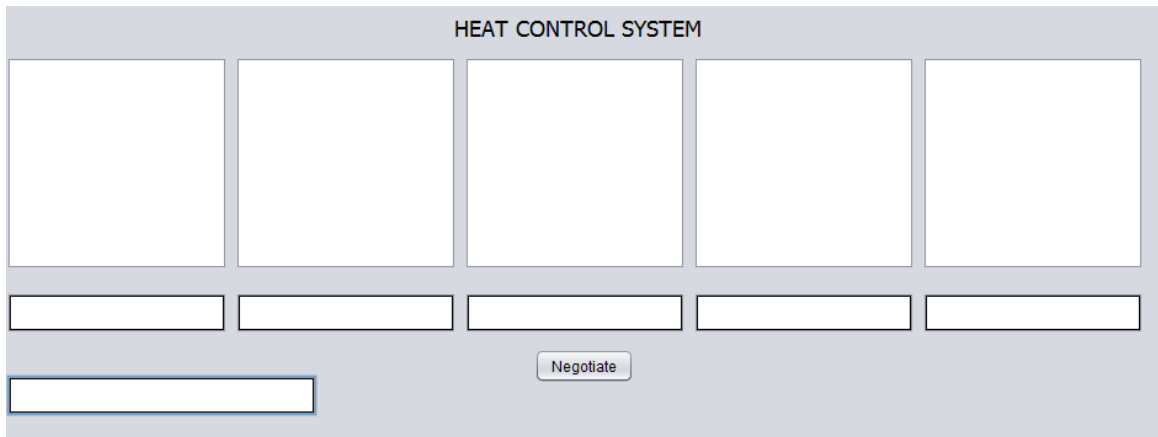


Figure 10.5.3 Heat Control Simulation

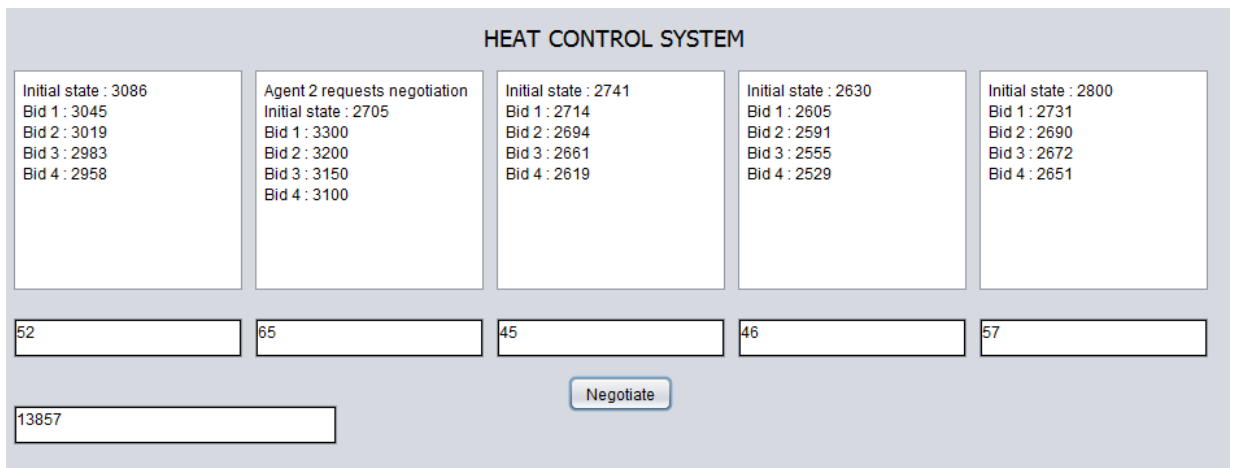


Figure 10.5.4 Heat Control Simulation during negotiation

10.5.3 SCREEN CAPTURE OF THE SIMULATION DURING THE ONE-BID NEGOTIATION



Figure 10.5.5 Initial state of the one bid negotiation

Agent 1 will give money to Agent 4	Agent 2 demands to Agent 1 You will cook during one day	Agent 3 demands to Agent 1 You will clean the house for two hours	Agent 4 demands to Agent 1 You will give money	Agent 5 demands to Agent 1 You will do the laundry of the week
<input type="text"/>	Request Bid 1 2.748	Request Bid 2 1.394	Request Bid 3 0.562	Request Bid 4 2.062
				Request Bid 5

Figure 10.5.6 Final state of the one bid negotiation

10.6 USER'S GUIDE

10.6.1 GENERAL INFORMATION

This user's guide will describe how to use the Java simulation to control the Smart Neighbourhood. To execute the simulation the user needs to open the Command Prompt and go to the directory where the .jar file is located. The, the user needs to write "java -jar SmartHouseApp.jar" in order to execute the file. Finally it will ask you to write the location of the "Ruleset.pl" file. To use it, the user needs to have the jpl library installed in his computer.

The simulation starts when the user clicks on the "Initial" button on the right side of the main simulation screen. After the simulation has been initialised, the devices can be turned on (the initial state for all the devices is off) and also the rules that we want to be serve can be activated by clicking on the rule name.

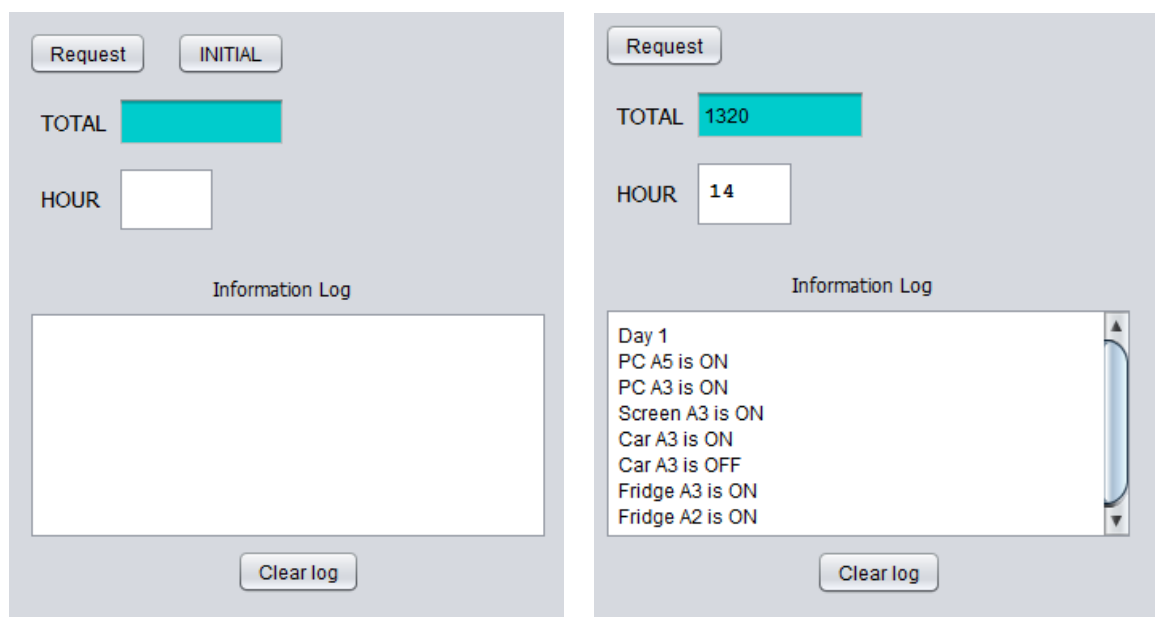


Figure 10.6.1. Information in the simulation

In the Figure 10.5. The information log, before pressing the “*Initial*” button and after 21 seconds of simulation. The “*Clear*” button is used to empty the log information area, although it does not start another new simulation. The blue area displays the total energy consumption of the neighbourhood and the white area displays the hour of the day during the simulation.



Figure 10.6.2. Device on and off

By clicking on the device number its state will change. The red colour states that the device is off and the green colour that the device is on. The two states are displayed in the Figure 10.6.

10.6.2 RULE-SET GUIDE

Below the information section, the rule control is found. That is where the user can select the amount of rules that he wants to have active in the Rule-set. The user can activate and de-activate a rule by clicking on it.



Figure 10.6.3. Rule control in the main simulation

10.6.3 HEAT CONTROL GUIDE

When the user clicks on the “*Initial*” button another Java window appears, that is the Java window entitled to control the heating of the house. This window can be seen in Figure 10.5.4.

Each time the user wants to start a negotiation he only has to press the “*Negotiate*” button. The results of a negotiation can be seen in Figure 10.5.4

10.6.4 ONE-BID SYSTEM GUIDE

The one-bid system guide starts when the user press the “*Request*” button in the Information log of the main simulation. After pressing the request button, the user needs to press the device that wants to turn on. Then the agents will place their bids and their requests, as it can be seen in Figure 10.6.4.

	Agent 2 demands to Agent 1 You will cook during one day	Agent 3 demands to Agent 1 You will clean the house for two hours	Agent 4 demands to Agent 1 You will give money	Agent 5 demands to Agent 1 You will do the laundry of the week
	Request Bid 1 2.748	Request Bid 2 1.394	Request Bid 3 0.562	Request Bid 4 2.062
				Request Bid 5

Figure 10.6.4. Rule control in the main simulation

The numbers displayed below each offer are the bid of the agents. Finally, the user needs to press the button below the agent that is requesting the negotiation (“*Request Bid*”) and device that wants to turn on again. The Figure 10.6.5 displays the information log with all the negotiation steps.

Agent 1 starts negotiation
Agent 1 negotiates with agent 4
Washing Machine A4 is OFF
Washing machine A1 is ON
Day 21

Clear

Figure 10.6.6. Information log during the negotiation

10.7 DATA FOR THE GRAPHS DISPLAYED

In this section, the data used to create the graphs that have been used to analyse the behaviour of the system will be displayed. All this data has been obtained through various simulations and stored into an Excel file to allow its manipulation.

10.7.1 DATA FOR THE HEAT CONTROL SYSTEM ANALYSIS

The Table 10.7 includes the final satisfaction of the agents after 100 successful negotiations and the total heat-energy consumption of the neighbourhood.

Satisfaction					Total energy
Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	consumption
57	51	53	54	59	14041
55	62	48	49	52	13883
51	57	67	46	49	13960
47	52	60	67	46	13975
45	46	53	58	65	13882
65	46	50	53	58	13982
56	65	45	50	53	13922
53	58	67	45	49	13975
48	51	59	65	45	13889
46	47	54	58	67	13969
65	47	49	53	59	13993
56	65	45	50	54	13951
51	55	65	45	51	13893
47	50	59	67	49	13983
65	45	55	58	45	13905
58	62	51	55	45	13960
51	54	45	50	65	13834
48	50	67	45	59	13930
45	45	59	67	55	13965
65	45	53	58	50	13960
56	65	47	53	45	13849

51	57	46	49	65	13878
45	53	67	45	61	13950
65	47	59	45	55	13965
58	45	53	65	51	13981
52	65	49	57	46	13944
49	58	46	53	67	14000
46	53	65	47	59	13964
62	49	55	46	54	13856
57	47	50	67	51	13987
52	65	48	60	45	13933
45	57	45	54	65	13860
65	50	45	51	57	13900
57	45	67	49	52	13921
52	65	60	45	46	13878
49	57	54	65	46	13954
45	52	51	58	65	13938
67	46	45	53	58	13936
58	45	65	49	53	13927
52	65	57	45	50	13918
47	57	51	65	46	13854
45	52	47	58	67	13939
65	46	46	53	58	13906
55	65	46	50	53	13918
51	56	65	45	50	13887
47	53	57	67	46	13956
45	46	52	60	65	13905
65	46	48	54	58	13956
55	65	46	50	53	13901
50	58	67	46	49	13961
45	53	59	67	45	13948
65	49	53	58	45	13947
57	45	50	53	67	13987
51	62	45	49	59	13875
46	56	67	46	55	13947

65	50	59	46	50	13917
57	45	54	67	45	13915
51	65	50	61	45	13984
46	57	46	55	65	13915
65	52	46	50	58	13955
57	47	67	45	54	13936
52	45	59	65	52	13972
45	65	53	56	46	13835
67	58	49	53	45	13974
57	53	45	47	65	13869
52	49	67	46	58	13967
49	45	60	65	53	13980
45	65	54	56	49	13908
65	56	50	52	45	13895
57	53	45	50	67	13994
53	49	65	46	58	13942
46	45	57	65	53	13860
46	65	54	56	49	13942
67	58	50	52	45	13992
58	53	45	47	65	13901
51	46	65	46	57	13834
47	67	58	46	52	13937
45	60	52	67	49	13993
65	54	45	60	45	13907
55	47	65	53	45	13850
51	45	58	50	67	13975
47	65	53	45	59	13922
45	56	49	65	55	13919
65	50	45	58	51	13917
58	45	67	53	47	13947
52	65	58	51	46	13976
48	57	54	48	65	13974
65	52	53	46	57	13999
56	45	47	65	53	13874

52	65	45	60	49	13958
47	58	67	56	45	13993
45	52	57	52	62	13905
67	47	52	49	57	13988
59	65	48	45	51	13888
53	57	46	65	46	13892
49	52	65	61	45	13968
45	45	56	55	65	13868
65	45	51	49	57	13879
57	67	48	47	53	13993
52	58	46	65	50	13954

Table 10.7.1. Satisfaction of the agents and total energy consumption

10.7.2 DATA FOR THE ONE-BID SYSTEM ANALYSIS

In the Table 10.7.2 the bids of the agents are placed. In each row there is an empty column which refers to the agent that is demanding to start the negotiation.

Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
	2.768	1.482	0.744	2.006
2.632	0.840	1.160	2.022	
2.406	0.000	1.624	1.784	
2.578	0.442	1.592	1.764	
2.202		0.542	2.746	1.312
2.798	1.940		1.328	0.528
2.664	2.166		1.612	0.902
0.378	2.368	2.190		1.534
1.780		0.602	2.660	1.430
0.446	0.000	0.000		1.178
	2.414	1.028	0.792	1.928
2.49	1.712		1.408	0.888
2.696	0.864	1.55	1.934	
2.356	2.216		1.424	0.978
	2.342	1.324	0.62	2.098

0.364	2.436	2.132		1.456
2.226		0.716	2.934	1.216
1.988		0.574	0	0
2.534	0.594	1.422	2.214	
	2.854	1.012	0.89	2.316
2.382	1.874		1.658	0.956
2.012		0.496	2.786	1.126
2.394	0.732	0	0	
2.866	0.604	1.408	2.334	
1.688		0.702	2.922	1.648
2.506	1.902		1.426	0.452
2.704	0.742	1.11	2.068	
0		0	2.532	1.38
0	1.89		1.656	0.54
	2.368	1.206	0.784	1.804
2.304	2.244		1.662	0.958
2.872	2.21		1.494	0.642
2.578	0.364	1.346	2.086	
1.974		0.584	2.696	1.404
0	0.53	0	2.306	
2.24		0.478	2.528	1.102
2.592	2.006		1.136	0.638
0	0.772	0	1.836	
2.572	1.688		1.692	0.722
2.572	1.688	2.142	1.692	0.722
2.148		0.808	2.59	1.17
2.33	1.668		1.39	0.508
2.804	0.738	1.208	2.278	
2.81	0.944	1.4	2.236	
2.81	0.944	1.4	2.236	1.662
2.546	2.038		1.606	0.89
0.584	2.782	2.24		1.286
	2.73	1.454	0.812	1.962

0.612	2.714	1.886		1.596
	2.862	1.44	0.602	2.128
0.93	2.752	2.186		1.634
0.576	2.644	2.198		1.296
	0	1.192	1.048	1.996
	2.656	1.544	0	2.078
2.462	0		0	0.706
0.53	2.338	2.236		1.29
0.642	2.464	1.896		1.684
	2.626	1.218	0.488	2.026
2.02		0.422	2.804	1.36
	2.704	1.15	0.904	2.264
2.338	1.92		0	0.536
0.544	2.866	2.13		1.356
2.746	0.696	1.11	1.936	
1.658		0.872	2.866	1.674
0	0		1.436	0.99
2.696	1.998		1.344	0.556
0	0	1.69	2.2	
2.348	0.646	1.35	2.016	
0	0		1.364	0.588
	2.356	1.27	0.726	1.856
	0	1.56	0.978	1.876
	0	1.56	0	1.876
	2.558	1.474	0	2.23
2.648	0.614	1.486	0	
0.432	2.584	2.184		1.408
2.07		0.59	2.674	1.156
0.744	2.87	2.1		1.258
2.712	1.788		1.562	0.552
0.922	0	2.004		1.468
0.42	0	1.802		1.238
0.444	2.56	1.862		1.54

	2.88	1.184	0.47	2.156
	0	1.128	0.696	2.296
1.916		0.806	2.996	1.278
0.92	2.636	1.952		1.588
1.854		0.882	2.484	0
1.788		0.626	2.756	1.658
0.47	2.802	2.028		1.124
2.104		0	2.876	0
2.75	1.922		1.174	0.78
	2.73	1.412	0.746	1.982
2.762	1.768		1.184	1.03
0.376	2.608	1.944		1.57
1.892		0.788	2.766	1.276
2.778	0.704	1.46	2.226	
	2.45	1.114	0.48	1.85
	2.582	0	0.814	0
1.888		0.56	2.858	1.108
	2.418	1.244	0.972	2.086
2.468	0	1.62	2.194	
	2.686	0	0.468	1.864
2.436	2.168		1.152	0.536

Table 10.7.2 Bids placed by the agents during 100 simulations

10.8 MINUTES OF PROJECT SUPERVISION MEETING